Contents

COMPUTER ARCHITECTURE

UNIT - I :

UNIT - II :

UNIT - III :

UNIT - IV :

UNIT - V :

UNIT

ORGANIZATION,
IEUMANN MODEL,
ON

: *(R.G.P.V., Dec. 2015)*

ronic machine that can solve
perations and presenting the
on of detailed step-by-step
ns, which cause a computer
gram.

l blocks that any computer
block. (R.G.P.V., June 2007)

onsists of five functionally
metic and logic, output and

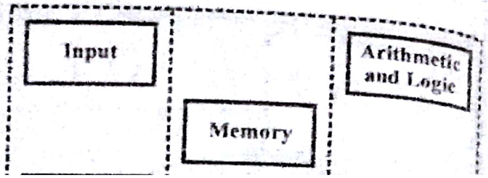
ation from human operators,
board of a video terminal, or
n lines.

stored in the memory for later
and logic circuitry to perform

...mined by a program stored in memory.

Output Control Unit – Finally the results are sent back to the outside world through the output unit. All of these actions are coordinated by the control unit.

It has been traditional to refer to the arithmetic and logic circuits in conjunction with the main control circuits as a ***central processing unit*** (CPU), or simply a ***processor***. Input and output equipment is us combined under the term *i* ***output unit*** (I/O).



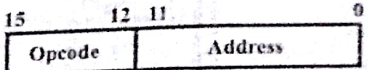
Q.3. Differentiate betw architecture.

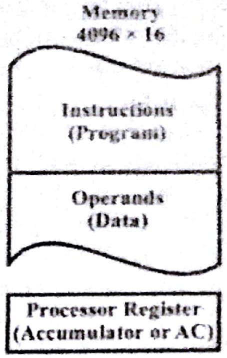
Ans. Differences betw architecture are given below in

S.No.	*Computer Organiz*
(i)	Computer organization the operational units an interconnections that re architectural specificati
(ii)	Hardware details which parent to the programm as the memory technol control signals and in between the computer pherals are referred to zational attributes.
(iii)	For example, an organiza the instruction by using multiply unit or by using nism that makes repeate the add unit of the syste

Q.4. Write short note on st

Ans. The simplest way to organize a computer is to have processor register and an instruction code format with two parts. The first part specifies the operation to be performed and the second specifies an address. The memory address tells the control where to find an operand in memory. This operand is read from memory and used as the data to be operated on together with the data stored in the processor register.





ganization

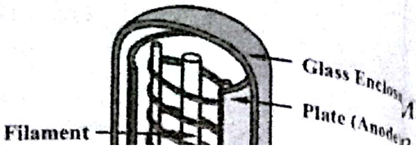
Instructions are stored in one memory unit with 4096 words $2^{12} = 4096$. If we store each e have available four bits for ssible operations and 12 bits rol reads a 16-bit instruction he 12-bit address part of the ta portion of memory. It then tion code. Computers that a name accumulator and label nory operand and the content

vices were invented that are roblems. All these computing which refer to the phases of es resulted in a small, cheap, nological development in the ements made to the hardware o the software technologies.

he first generation computers used the vacuum tubes technology for calculation as well as for storage and control purposes. So, these computers are also called as vacuum tubes or hermionic valves based machines. A vacuum tube, as shown in fig. 1.3, was fragile glass device, which used filaments inside it. The filaments when eated generate electrons, which eventually help in the amplification and

deamplification of electronic signals. These vacuum tube computers could perform computations in milliseconds. The memory of these computers was
constructed using electroma
and all data and instructions
the system from punched
instructions were written in
assembly languages becaus
programming languages wer
much later. These first
computers were mainly used
computations. Some examp
computers are ENIAC, EDV
UNIVAC I and IBM 701.

(ii) Second Gen
device, called transistor, v
Bardeen, Willian Shockley
Prize. A transistor, as show
to increase the power of t
original signal. It has thre
collector (C). The base o
needed to be amplified, is s
is generally a small flow o
the input gate for the tra
collect the amplified sign
gate for emitting the ampli
generation computers were
tubes to build the basic lo
computers were smaller, f
vacuum tubes. Transistor
light weight electronic d
required very less power
operation. These characteri
of transistors made t
generation computers mor. powerful, more reliable, less expensive, smaller and cooler to operate than the first generation computers. Printers, secondary storage and operating system technology were also invented during this era.



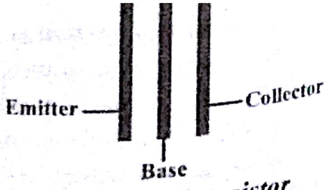


Fig. 1.4 A Transistor

The memory of these computers was composed of magnetic cores. Magnetic disk and magnetic tape were the main secondary storage media used in second generation computers. Punched cards were still popular and widely
to these computers. Another
mputers was the replacement
age. Assembly language is a
o use simple English words–
uctions in a program. The
multiple jobs to be batched
transition from one job to
concept helped in reducing
resulting in faster processing,
nd generation computers. In
ge of computers was seen in
sing applications like payroll,
ng. Some examples of second
IBM 7090.

n 1958, Jack St. Clair Kilby
cuit. The major characteristic
the use of Integrated Circuits
veral electronic components
on a single chip of silicon,
ponents. The IC technology
y because it made it possible
into very small (less than 5
' (see fig. 1.5). Initially, the
nents. This technology was
e advancement in technology
tegrate up to about hundred

components manually. Hence, third generation computers were more powerful, more reliable, less expensive, smaller and cooler to operate than the second generation computers.

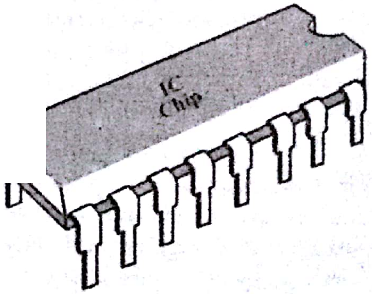


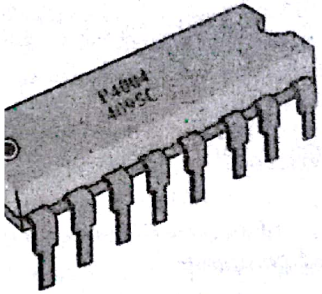
Fig. 1.5 An Integrated Circuit

Parallel advancements in storage technologies allowed the construction
larger magnetic cores based random access memory and larger capacity magne
disks and magnetic tapes. Hence, the third generation computers typically h
few megabytes (less than 5 megabytes) of main memory and magnetic dis
capable of storing few tens o
front, the third generation sa
programming languages, time
from hardware and the creati
and COBOL, which were the
those days, were standardized
in 1966 and 1968 respectivel
ANSI COBOL. The idea wa
program writing, a FORTRAN
with an ANSI FORTRAN or
high-level programming lang
period. Notable among these v
Thomas Kurtz of Dartmouth
operating system. Time-shari
number of users to directly ac
that each user gets the illusi
accomplished by having a la
online terminals simultaneou
computer manufacturers sold t
and did not charge separately f
the situation changed in 1969
to price their hardware and
software from hardware gave
their need and value. The dev
took place during the third ge
1960s were mainframe system
purchase and use. The first c
was introduced in 1965 by Di
the tremendous demand for sm
and by 1971, there were more
the minicomputer market. Som
395, B6500, IBM 370, PDP 11

(iv) Fourth Gener

electronic components packe on a silicon chip doubled each year after 19
This progress soon led to the era of large scale integration (LSI) when it w
possible to integrate over 30000 electronic components on a single chi
followed by very large scale integration (VLSI) when it was possible to integra
about one million electronic components on a single chip. As a result, th
manufacturers were able to reduce the size of the computers and made the

cheaper as compared to the other generation of computers. The progress in
LSI and VLSI technologies led to the development of microprocessor. A
microprocessor incorporates various
of a computer--such as CPU,



ig. 1.6 The Intel P4004 Microprocessor Chip

memories were replaced by
ı access memories with very
ecame cheaper, smaller and
: tapes, floppy disks became
cant development during the
speed computer networking,
d together, to enable them to
ular for connecting several
an organization and WANs
at larger distances. This gave
ms. Some of the examples of
/AT, Apple and CRAY-1.

he different types of modern
th generation computers. The
arge scale integration (ULSI)
components to be fabricated
creasing the power and speed
rimary and secondary storage
neration computers are faster,
urth generation computers.

igital computer. Explain its
.V., June 2005, 2008, 2011)

Or

Describe the Von-Neumann model and explain the functioning of its components. (R.G.P.V., June 2012, 2013)

Or

Draw and explain Von-Neumann model of computer and explain its subsystems. (R.G.P.V., June 2014)

Or

Explain Von-Neumann model for computation. (R.G.P.V., Dec. 201

Or

Describe Von-Neumann model with the help of diagram.
(R.G.P.V., June 201

What is Von-Neumann
systems.

Draw Von-Neumann m
of computer.

Ans. Von-Neumann and l
the design of a new stored-p
1946, known as the IAS
Princeton Institute for Adv
IAS computer is the prototyp
general purpose computers. T
of the IAS computer is sh
consists of –

(i) A main memor
(ii) An arithmetic an
(iii) A control unit
causes them to be executed.
(iv) Input and outp

Q.7. Draw Von-Neuman
bottleneck ?

***Ans.* Von-Neumann Ar**

Von-Neumann Bottlene
affected by other factors besid
to move instructions and data
lesser extent, the time required
It typically takes the CPU ab
than from one of its internal
since the first electronic co
designers to develop memory devices and processor-memory interface circui
that are fast enough to keep up with the fastest microprocessors. Indeed th
CPU-M speed disparity has become such a feature of standard (Von-Neumann
computers that is sometimes referred to as the *Von-Neumann bottleneck*. RISC
computers usually limit access to main memory to a few load and stor
instructions, other instructions, including all data processing and program

ontrol instructions, must have their operands in CPU registers. This so-called *oad-store architecture* is intended to reduce the impact of the Von-Neumann ottleneck by reducing the total number of the memory accesses made by the CPU. Caches directly address the Von-Neumann bottleneck by providing the CPU with fast single-cycle access to its external memory.

ann model.
(R.G.P.V., June 2016)

model are as follows –
l in a single read write memory.
addressable by location without

tial fashion (unless explicitly

Von-Neumann and Harvard
(R.G.P.V., Dec. 2009)

gram and data are stored in the
rmation-handling subsystem. In
stored and handled by different
etween the two architectures.
ilt in 1944 and for which the
task and the data-handling task
rent storage technologies. Today,
ann architecture because of the
ng, and operating one memory

ain embedded applications where
requirements are such that the
tional advantages. Under certain
ch faster than a Von-Neumann
ontend for the same information
mutable read-only memory can

nctions.

Ans. The CPU contains three major parts, as depicted in fig. 1.8.

(i) The register set which stores intermediate data used during the execution of the instructions.

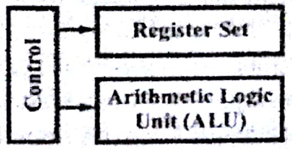


Fig. 1.8 Major Parts of CPU

(ii) The arithmetic logic unit that performs the required mic operations for executing the instructions.

(iii) The control unit that supervises the transfer of information amo the registers and instructs the ALU as to which operation to perform.

The CPU performs a var
dictated by the type of instruct
which are incorporated in th
is defined as the computer st
which uses machine langua
addressing modes, the instru
CPU registers. The boundar
programmers see the same n
instruction set.

From the designer's poi
the specifications for the des
large part involves selectin
instructions. The computer p
language must be aware of t
data supported by the instr
performs.

REGISTER ORGAN

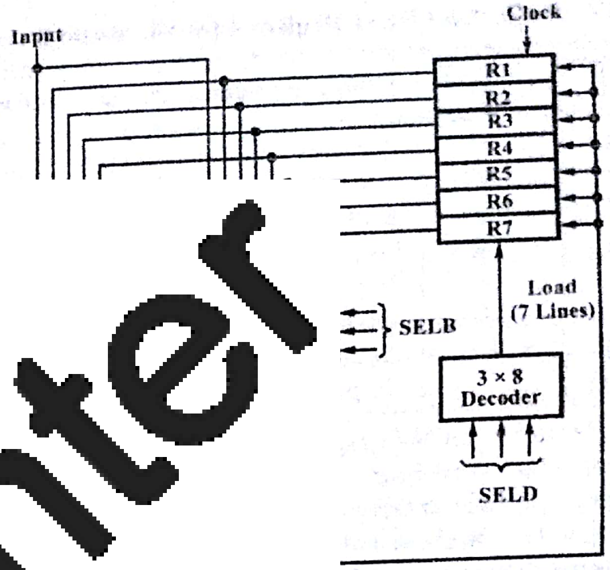
Q.11. What is register

Ans. A register is a gr
storing one bit of informatio

Q.12. Write short note

Ans. Memory locations
products, pointers, counters,
access is the time consuming
time consuming. It is more effi
values in processor registers.
only for performing various m
For seven CPU register, a bu

In fig. 1.9, the output of each register is connected to two multiplexers form the two buses A and B. For a specific bus the selection times in eac multiplexer choose one register or the input data. The A and B buses forms th inputs to a common arithmetic logic unit. The operation choose in the ALI determines the arithmetic or logic microoperation that is to be performe Microoperation result is available for output data and also goes into the input



ster which gets the information
one of the register load inputs,
data in the output bus and the

s system directs the information
ting the various components in

isters. ***(R.G.P.V., June 2010)***

consecutive memory locations
The control reads an instruction
s it. It then continues by reading
tes it and so on. This type of
lculate the address of the next
truction is completed. It is also
necessary to provide a register in the control unit for storing the instruction code after it is read from memory. The computer needs processor registers for manipulating data and a register for holding a memory address. The register are listed in table 1.1 together with a brief description of their function and the number of bits that they contain.

Table 1.1 List of Registers for the Basic Computer

Register Symbol	Number of Bits	Register Name	Function
DR	16	Data register	Holds memory operand
AR	12	Addres	[redacted]
AC	16	Accum	[redacted]
IR	16	Instruct	[redacted]
PC	12	Prograi	[redacted]
TR	16	Tempoı	[redacted]
INPR	8	Input r	[redacted]
OUTR	8	Output	[redacted]

The memory unit has a capa
bits. Twelve bits of an instructic
an operand. This leaves three bit
a bit to specify a direct or indire
operand read from memory. The a
processing register. The instru
instruction register (IR). The te
temporary data during the proces
12 bits since this is the width c
(PC) also has 12 bits and it holds
from memory after the current
(INPR) receives an 8-bit charac
(OUTR) holds an 8-bit character
[redacted]

Q.14. Write down different
the general register organizatioı

Ans. Refer to Q.13 and Q.1

Q.15. Define the accumula

Write the function of accun
(R.G.P.

Ans. The accumulator is a re
the execution of an instruction, and receives the result of most of the arithmetic and logical operations.

Q.16. Explain program counter. ***(R.G.P.V., June 2008, Dec. 2011)***

Or

Write the function of program counter in computer system.
(R.G.P.V., Dec. 2005, May/June 2006, Dec. 2013)

Ans. The program counter keeps track of the address of the instruction which is to be executed next. Therefore, it holds the address of the memory location which contains the next instruction to be fetched from the memory. After an instruction has been fetched, its content is automatically incremented assuming that instructions are normally executed sequentially. For a jump
[redacted]
n jumps to the memory location
ecuted next.

r. (R.G.P.V., June 2008)

gister in computer system.
, May/June 2006, Dec. 2013)

ldress of the instruction or data
essing unit (CPU) transfers the
m counter (PC) to the memory
it is transmitted to the memory
called simply address register.

egister in computer system.
, May/June 2006, Dec. 2013)

(R.G.P.V., June 2008)

n being executed. This register
computers have two instruction
t instruction while the execution

ry data register in computer
(R.G.P.V., Dec. 2005)

(R.G.P.V., June 2008)

struction code or data received
to data bus. The data that are
ster until the write operation is
CPU to the memory or from the
lata register.

BLEMS

Prob.1. The following program is stored in the memory unit of the basic computer. Show the contents of the AC, PC and IR (in hexadecimal), at the end, after each instruction is executed. All numbers listed below are in hexadecimal –

Location	Instruction
010	CLA
011	ADD 016
012	
013	
014	
015	
016	
017	

[redacted]

Sol.

Location	Instruction	Con
010	CLA	
011	ADD 016	
012	BUN 014	
013	HLT	
014	AND 017	
015	BUN 013	
016	C1A5	
017	93C6	

$(C1A5)_{16}$ = 1100 0001 10

$(93C6)_{16}$ = 1001 0011 11

1000 0001 10

Prob.2. ***A digital computer***

8 bits each. The bus is constru

(i) How many selecti

(ii) What size of mult

(iii) How many multip

(R.G.P.V., Dec. 2009)

Sol. (i) There are total $2^3 = 8$ registers.

Therefore, 3 selection lines are required to select one of 8 registers.

(ii) 8 × 1 multiplexer is needed.

(iii) 8 multiplexers, one for each bit of the registers.

Prob.3. ***A digital computer has a common bus system for 16 registers of 32 bits each. The bus is constructed with multiplexers –***

(i) How many selection inputs are there in each multiplexer ?

(ii) What size of multiplexer is needed ?

re in the bus ?

(R.G.P.V., Dec. 2011, 2012)

choose one of 16 registers.

t of the registers.

IEMORY TRANSFERS

fer language'.

(R.G.P.V., June 2004, 2008)

ıbolic notations used to describe s. The term 'register transfer' uits which can perform a stated eration of the same or another from programmers, who apply r transfer language (RTL) is a rganization of digital computers ansfer language is a system for n sequences among the registers uage is believed to be as simple ong to memorize. It can also be igital systems.

nsfer and write down the basic

egister to another is represented operator. The statement

$$R2 \leftarrow R1$$

shows a transfer of the content of register R1 into register R2. The above expression designates that the content of R2 is replaced by the content of R1. However, after the transfer, the content of source register R1 does not change.

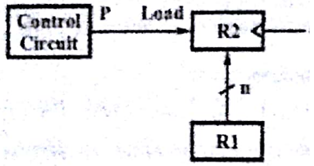
The statement that specifies a register transfer implies that circuits are available from the outputs of the source register to the inputs of the destinatio register and that the destination register has a parallel load capability. Usuall it is desired that the transfer may take place only under a predetermined cont condition. This can be denot

If

where P is a control signal p convenient to separate the con by specifying a ***control func*** which is equal to 1 or 0. In th given below –

P : R2 ← R1

This indicates that transf only if P = 1.



(a) Block Diagram

Fig. 1.10 Tran

Fig. 1.10 depicts the blo R2. Register R2 has a load in Here, it is assumed that the c clock as the one applied to th

It is shown in timing dia control section by the rising transition of the clock at time data inputs of R2 are loaded in back to 0; otherwise, the transfe while P remains active. Table transfer notation.

Table 1.2 Basic

Symbol	*Description*	*Examples*
Letters (and numerals)	Denotes a register	MAR, R2
Parentheses ( )	Denotes a part of register	R2(0 - 7), R2(L)
Arrow ←	Denotes transfer of information	R2 ← R1
Comma,	Separates two microoperations	R2 ← R1, R1 ← R2

Here, registers are represented by capital letters, and numerals may follow the letters. To represent a part of a register by specifying the range of bits or by giving a symbol name to a portion of a register, parentheses are used. of information and the direction of transfer. A comma ch are executed at the same of two registers during one oted by the statement as given

R2

ops, are used to make this

P.V., Dec. 2008, June 2013)

le which is equal to 1 or 0. In 1 as given below –

l be executed by the hardware

P.V., Dec. 2008, June 2013)

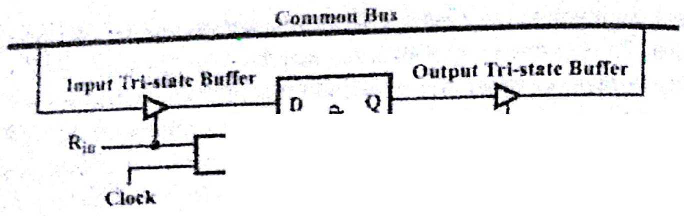
th three state gates instead of it that shows three states. Two and 0 as in a conventional gate high impedance state behaves itput is disconnected and does may perform any conventional ne of most commonly used in graphic symbol of a three state

A if C = 1
dance if C = 0

Q.24. Draw and explain the implementation of 1-bit register.

(R.G.P.V., Dec. 2014)

Ans. The implementation of one bit register is shown in fig. 1.12. The edge triggered D flip-flop is stored the 1-bit data, which is connected to the common bus through tri-state buffers.



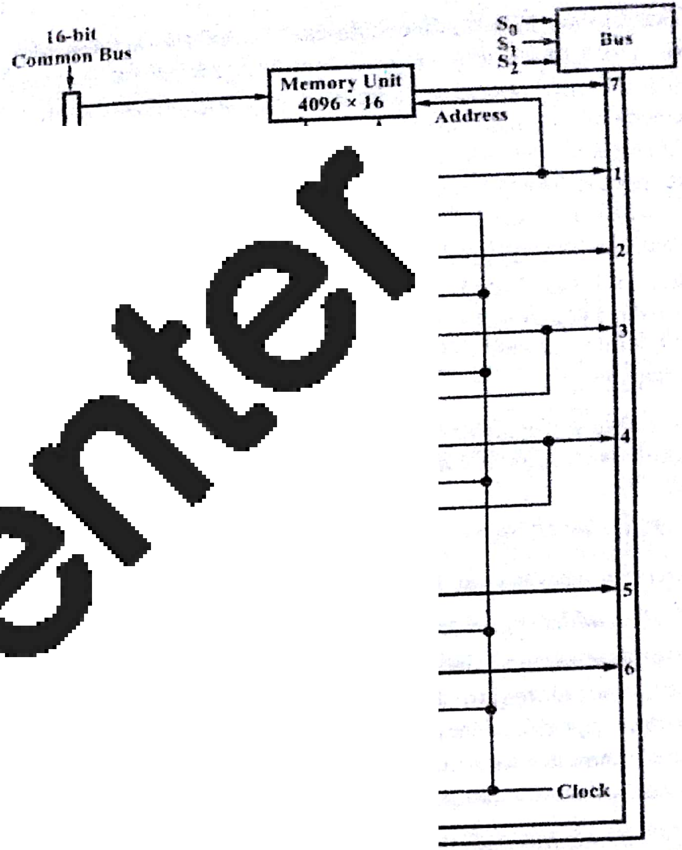
Fi

The input D and outpu
and output tri-state buffer re
enables the input tri-state bu
the D flip-flop in synchroniz
AND gate (see fig. 1.12). The
Q output of the D flip-flop o
state buffer.

The bus contention pro
buffers are active at a time. T
buffer may be in the active s

Q.25. Draw and expl
connections between the reg

Ans. The basic compute
and eight registers. Paths m
register to another and betw
made between the outputs o
the number of wires will be
information in a system w
Fig. 1.13 illustrates the conn
computer to a common bus.
connected to the common bus
chosen for the bus lines is de
variables S_2, S_1 and S_0. The
equivalent of the required bin
output of IR is 5. The 16-bit
$S_1 S_0 = 101$ because this is th
bus are connected to the data inputs of the memory and to the inputs of eac
register. During the next clock pulse transition, the particular register whose load (LD) input is enabled receives the data from the bus. If the memory write input is activated then it receives the data from the bus. If the read input is activated and $S_2S_1S_0 = 111$ then the memory places its 16-bit output onto the bus.



cted to a Common Bus

each since they hold a memory
to 0's when the contents of AR
n AR or PC receive information
are transferred into the register.
each. The input register (INPR)
each and communicate with the
ines of the common bus receive
gisters. Five registers have three
) and LD (load). The increment
operation is obtained by enabling the count input of the counter. Two registers have only a load input.

The input data and output data of the memory are connected to the common bus, however, the memory address is connected to AR. Thus, AR must always be used to specify a memory address. By employing a single register for the

address, we remove the need for an address bus that would have been required otherwise. The content of any register can be specified for the memory data input during a write operation. Likewise, any register can receive the data from memory after a read operation except AC. The 16 inputs of AC come
from an adder and logic circuit t
16-bit inputs come from the data
come from the outputs of AC
microoperations like shift AC an
AC are utilized for arithmetic and
add DR to AC. The result of the c
out of the operation is sent to flip
the input register.

Q.26. Explain the different
Explain how these registers are

Ans. **Types of Registers –**

Connection between the Re

Q.27. Why address and data

Ans. The address and data b
pins. Since we do not require add
a common bus for address and
address first, when the location
data with that selected location.

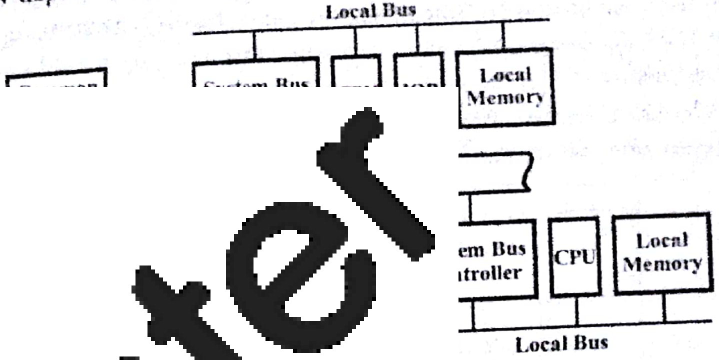
Q.28. Write explanatory not

Explain common bus syste

Ans. A typical computer sys
to facilitate the transfer of informa
a number of internal buses for
registers and ALU. A bus which co
system, such as CPUs, IOPs, and

In a shared memory multipro... system, the processors request access to common memory or other common resources through the system bus. If no other processor is currently using the bus, the requesting processor may be granted access immediately. However, if another processor is currently utilizing the system bus, the requesting processor must wait. In addition, other processors may request the system bus at the same time. Then, arbitration

must be performed to resolve this multiple contention for the shared sources. Fig. 1.14 depicts the system bus structure for multiprocessors.



Multiprocessors

100 signal lines. Functionally,
id control. Furthermore, there
ver to the components. Data
ween processors and common
a multiple of 8, with 16 and 32
to recognise a memory address
it or output ports. The number
sible memory capacity in the
can access up to 2^{24} (16 mega)
e terminated with three-state
g the transfer of data in either
om process to memory.

cur in two ways – synchronous
nsfer between units, the control
he validity of data and address
tions to be performed. Typical
ry read and write, acknowledge
ignals such as bus request and
es.

bus system that use multiplex
k registers of n-bits each to produce an n-line common bus.
(R.G.P.V., Dec. 2008, June 2012)

Or

Draw and explain the bus structure for the data transfer between registers and the common bus. ***(R.G.P.V., June 2013)***

Ans. A bus system will multiplex k registers of n-bits each to produce an n-line common bus. The number of multiplexers needed to construct the bus is equal to n, the number of bits in each register. The size of each multiplexer must be k × 1 since it multiplexes k data lines. For example, a common bus for eight registers of 16 bits each req
bus. Each multiplexer must have
to multiplex one significant bit i

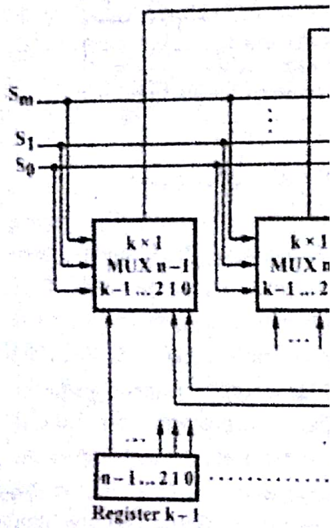
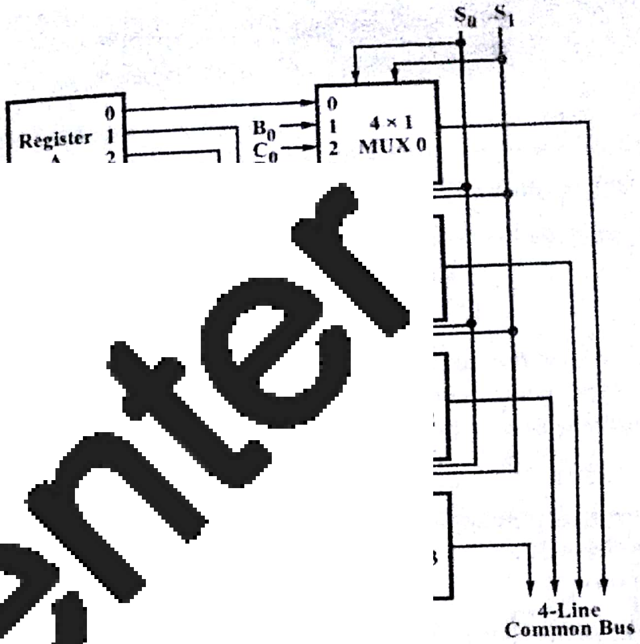


Fig. 1.15 B

The construction of a bus
Each register has n bits, numbe
k × 1 multiplexers each having k
selection lines will be such that

Q.30. Draw a common bus
multiplexers.

Ans. A bus structure compo
bit of a register, through which binary information is transferred one at a time. Control signals determine which register is chosen by the bus during each particular register transfer. The multiplexers select the source register whose binary information is placed on the bus. The construction of a bus system for four registers is illustrated in fig. 1.16.



ır Register

mbered from 0 to 3. The bus
ining four data inputs from 0
. 1.16, we use labels to show
the inputs of the multiplexers
the same significant position
ts of one multiplexer to form
the four 0 bits of the register,
r and so on. The two selection
nputs of all four multiplexers.
egister and transfer them into
of
00
is
of
re
...multiplexers.

Table 1.3 Function Table for Bus

S_1	S_0	*Register Selected*
0	0	A
0	1	B
1	0	C
1	1	D

In a similar way, register B is chosen if $S_1S_0 = 01$ and so on. Table 1.3 shows the register which is selected by the bus for each of the four possible binary value of the selection lines.

NUMERICAL PROBLEMS

Prob.4. Represent the following conditional control statement by two register transfer statements w

If (P = 1) then (R_1 ←

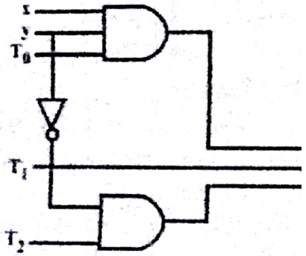
Sol. P : R

P'Q : I

Prob.5. Show the hardw include the logic gates for the binary counter with a count

$xyT_0 + T_1 + y'T_2$

Sol. The hardware implem is shown in fig. 1.17.



Prob.6. Show the block di following register transfer stat

$yT_2 : R_2 \leftarrow R_1$, I

Sol. Fig. 1.18 shows the bl the given register transfer state

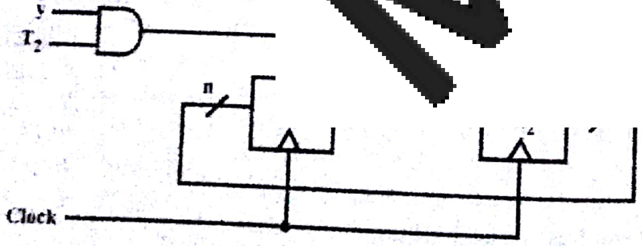


Fig. 1.18

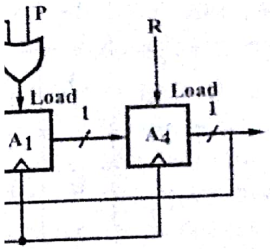
Prob.7. Design a hardware circuit by using common bus architecture to implement the following register transfer languages –

$P : A_1 \leftarrow A_2$

$Q : A_2 \leftarrow A_3$

$R : A_4 \leftarrow A_1$

; architecture to implement



nsfer statements for two 4-

ontent of R2 is added to the nsferred to R1 if x = 0. Draw ition of the two statements. 4-bit adder and a quadruple to R1. In the diagram show nputs of the multiplexer and

ove given statements is shown

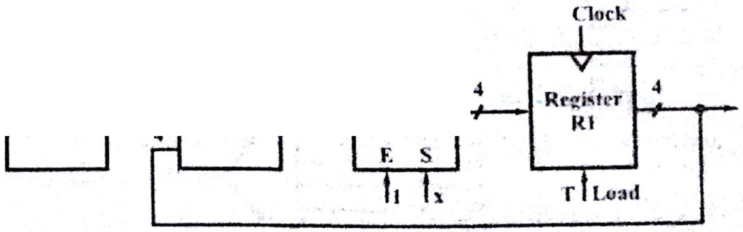


Fig. 1.20

Prob.9. Show the hardware implementation for the following statements. The registers are 4-bit in length.

$T_0 : A \leftarrow R_0,$ $T_1 : A \leftarrow R_1$

$T_2 : A \leftarrow R_2,$

Sol. The conditions are

table –

T_0	T_1
0	0
1	0
0	1
0	0
0	0

The condition statements a

$S_1 = T_2 + T_3$

$S_0 = T_1 + T_3$

Load $= T_0 + T_1$

The block diagram showin

transfer is shown in fig. 1.21.

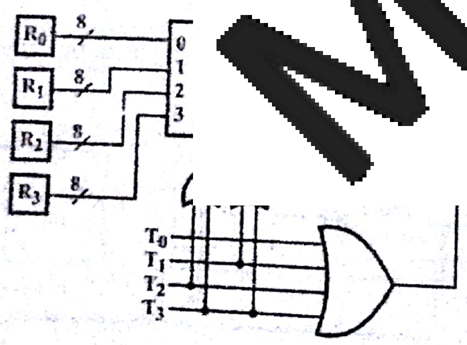


Fig. 1.21

ARITHMETIC, LOGIC AND SHIFT MICROOPERATIONS, ARITHMETIC LOGIC SHIFT UNIT

(R.G.P.V., June 2010)

R.G.P.V., Dec. 2007, 2011)

2004, 2005, Dec. 2008, 2009, June 2012, 2013)

erformed on the information
ined after the operation may
ter or may be put into another
mples of microoperations. A
rooperations increment and
the shift right and shift left
nal hardware organization is

ir function.

performed on the binary

of microoperations.

erations can be specified by
ocedure generally involves a
enient to adopt an appropriate
ers between registers and the
associated with the transfers.
ination provides an organized
sequences in registers and the

microoperations ?

our categories –

(i) Register transfer microoperations transfer binary information from one register to another.

(ii) Arithmetic microoperations perform arithmetic operations on numeric data stored in registers.

(iii) Logic microoperations perform bit manipulation operations o non-numeric data stored in registers.

(iv) Shift microoperations perform shift operations on data store in registers.

Q.33. Explain arithmeti

Ans. Addition, subtractio arithmetic microoperations. An

It states that the content register R2 and the sum transfe of this statement needs three r the addition operation. Other table 1.4.

Table 1.4 A

Symbolic Designation	
R3 ← R1 + R2	Cont
R3 ← R1 − R2	Cont
R2 ← $\overline{R2}$	Com
R2 ← $\overline{R2}$ + 1	2's c
R3 ← R1 + $\overline{R2}$ + 1	R1 pl
R1 ← R1 + 1	Incre
R1 ← R1 − 1	Decr

In general, subtraction i addition. Instead of using the the following statement –

R

$\overline{R2}$ denotes the 1's comp 2's complement. Adding the c equivalent to R1 – R2. Increment and decrement microoperations are symbolize by plus-one and minus-one operations.

In most computers, the multiplication operation is implemented with sequence of add and shift microoperations, and division is implemented with sequence of subtract and shift microoperations.

Q.34. What do you understand by logic microoperations ? Enlist the various logic microoperations.

Ans. Logic microoperations are the microoperations which specify binary operations for strings of bits stored in registers. In these operations, each bit considered separately and treated like binary variables. As an the contents of two registers

cuted on the individual bits of = 1. Special symbols are used mplement to distinguish them press Boolean functions. The eration and the symbol ∧ to lement microoperation is the on top of the symbol which o differentiate between a logic on by using different symbols. ls is to be able to distinguish hmetic plus, from a logic OR icrooperation, it will represent rol or Boolean function, it will statement

– R5 ∨ R6

veen two binary variables of a enotes an add microoperation. ol ∨ between registers R5 and

e performed with two binary sible truth tables obtained with . In this table, each of the 16 f one possible Boolean function

ons of Two Variables

												F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

The first column of table 1.6 represents these 16 Boolean functions two variables x and y in algebraic form. The 16 logic microoperations obtained from these functions by replacing variable x by the binary conte of register A and variable y by the binary content of register B. Boole functions listed in first colur

two binary variables x and

column denote a relationshi

and B.

Table 1.6 S

Boolean Function
$F_0 = 0$
$F_1 = xy$
$F_2 = xy'$
$F_3 = x$
$F_4 = x'y$
$F_5 = y$
$F_6 = x \oplus y$
$F_7 = x + y$
$F_8 = (x + y)'$
$F_9 = (x \oplus y)'$
$F_{10} = y'$
$F_{11} = x + y'$
$F_{12} = x'$
$F_{13} = x' + y$
$F_{14} = (xy)'$
$F_{15} = 1$

Q.35. What are the vari

Write short note on shi

Ans. Shift microoperati contents of a register can be types of shifts –

(i) Logical shift

(ii) Circular shift

(iii) Arithmetic shift.

(i) Logical Shift – A logical shift is one which transfers 0 throug the serial input. The symbols ***shl*** and ***shr*** are used for logical shift-left an

shift-right microoperations. As example –

$$R1 \leftarrow shl\ R1$$

$$R2 \leftarrow shr\ R2$$

hift to the left of the content

the content of register R2.

nd position through the serial

or rotate operation circulates

without loss of information.

the shift register to its serial

circular shift left and right,

lic notation for the shift

rations

ription
egister R
register R
iift-left register R
hift-right register R
shift-left R
shift-right R

shift microoperation shifts a

ithmetic shift-left multiplies a

shift-right divides the number

unaltered since the sign of the

or divided by 2.

ift microoperations.

(R.G.P.V., Dec. 2017)

Ans. Refer to Q.34 and Q.35.

Q.37. Explain different microoperations with example.

(R.G.P.V., June 2016)

Ans. Refer to Q.33, Q.34 and Q.35.

Q.38. Explain arithmetic logic unit in detail.

Ans. Instead of using the individual registers to perform t microoperations directly, computer systems uses various storage registe connected to common ALU. The data of particular registers are kept in t inputs of the common ALU to perform a microoperation. Then, the A performs an operation and it
ALU is a combinational cir
from the source registers th
can be performed during
microoperations are perform
overall ALU.

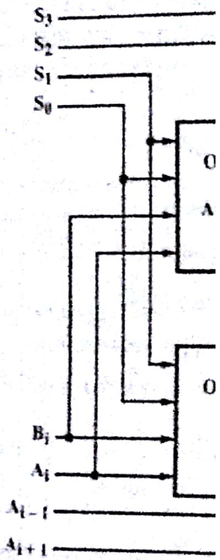


Fig. 1.22 illustrates one
figure, the arithmetic, logic a
common selection variables.
and B_i are given to both the a
S_i are used to select a micr
chooses between an arithme
multiplexer, the data are selected with inputs S_2 and S_3. The other two da inputs to the multiplexer are A_{i-1} for shift right operation and A_{i+1} for shi left operation. The output carry C_{i+1} of a given arithmetic stage must b connected to the input carry C_i of the next stage in sequence. The input car of the first stage is the input carry C_{in} which provides a selection variable f

the arithmetic operations. One stage of ALU unit shown in fig. 1.22 provides eight arithmetic operation, four logic operations and two shift operations.

Table 1.8 Operations of the ALU

Operation Select		*Function*
		Transfer A
		Increment A
		Addition
		Add with carry
		Subtract with borrow
		Subtraction
		Decrement A
		Transfer A
		AND
		OR
		XOR
		Complement A
		Shift Right A into F
		Shift left A into F

ration ? Explain any four
gic unit with its function
(R.G.P.V., May 2018)

nicrooperations cannot be

that will perform the above

(R.G.P.V., May/June 2006)

Here, PC cannot provide address to memory. Address must be transferred to AR first.

AR ← PC

IR ← M[AR]

(ii) AC ← AC + TR

Here, add operation must be done with DR. Transfer TR to DR first

DR ← TR

AC ← AC + DR

(iii) DR ← DR + A

Here, result of addition

AC its content must be store

AC ← DR, DR

AC ← AC + DF

AC ← DR, DR

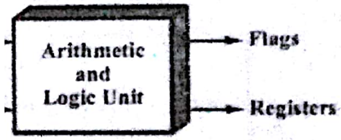
UNIT

IT, FIXED POINT
ESENTATION, SIGN-
MENT AND RANGE

(ALU).

omputer system is the place
; place during the processing
lculations performed and all
and instructions, stored in the
l as and when needed, to the
ults generated in the ALU are
: until needed at a later time.
again to storage many times
rithmetic and logic operations
ict, multiply, divide and logic
to or greater than.

.U is interconnected with the
e ALU in registers and the
hese registers are temporary
onnected by signal paths to
ılt of an operation. The flag



of the ALU.

Fig. 2.1 ALU I/Ps and O/Ps

Q.2. Write down the functions performed by ALU.

Ans. ALU (arithmetic and logic unit) is the part of the CPU which performs arithmetic and logic operations. Generally, an ALU performs the following

arithmetic and logic operations –

- (i) Addition
- (ii) Subtraction
- (iii) Multiplication
- (iv) Division
- (v) Logical AND
- (vi) Logical OR
- (vii) Logical exclusive-OR
- (viii) Complement (logical NOT)
- (ix) Increment (add
- (xi) Left or right shi

left or right by one bit)

- (xii) Clear (the conter

ALU does not perform oth

logarithmic, trigonometric and

performed by special purpose

Modern microprocessors con

(i.e., an on-chip FPU).

Q.3. Discuss the design o

Take an example and ex

Ans. Functionally, an A

arithmetic unit and the logic un

operations such as addition,

Usually, the operands involve

cases, however, an arithmetic

(BCD) numbers and floating-

necessary electronics to manip

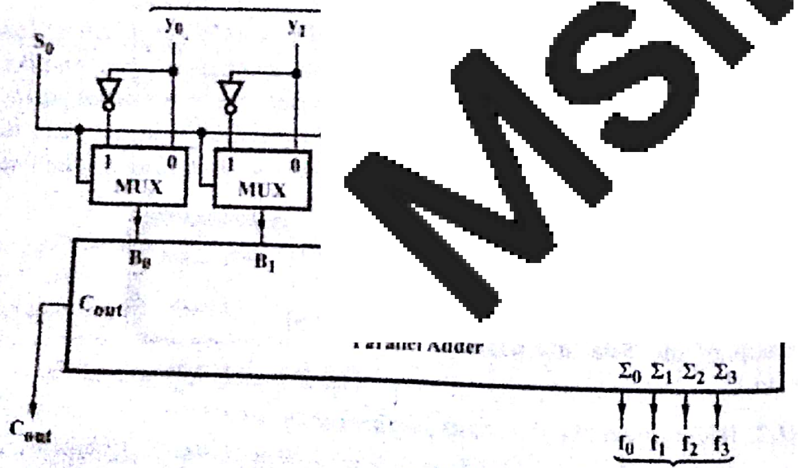


Fig. 2.2 Organization of an Arithmetic Unit

logic unit contains hardware elements that perform typical operations such as Boolean NOT and OR. Here, the design of a simple ALU using typical combinational elements such as gates, multiplexers, and a 4-bit parallel adder is discussed. For this approach, first an arithmetic unit and a logic unit are designed separately, then they are combined to obtain an ALU.

unit as shown in fig. 2.2, is

arallel adder. The multiplexer

rallel adder. In particular, if

he selection input (S_0) also

lts –

s shown in fig. 2.3, is designed.

utput G = X AND Y; otherwise

ean operations, other operations

Boolean identities –

ed by using additional hardware

y the arithmetic and logic units

s as shown in fig. 2.4.

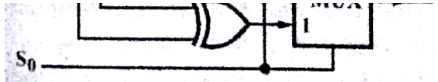


Fig. 2.3 Organization of a Logic Unit

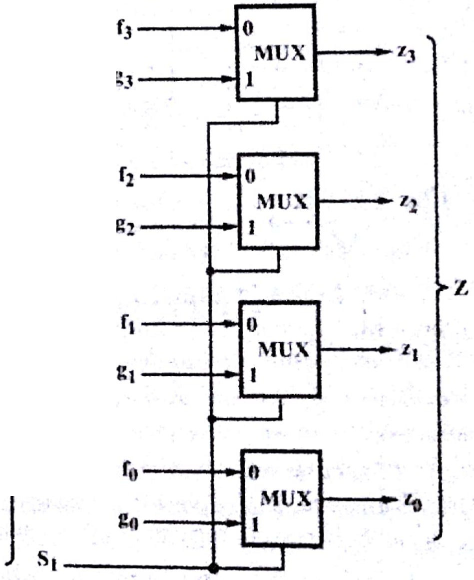


Fig. 2.4 Combining the Outputs Generated by the Arithmetic and Logic Units

From this organization it can be seen that when the select line $S_I = 1$, multiplexers select outputs generated by the logic unit; otherwise, the outp of arithmetic unit are selected. The select line, S_1, is referred to as the m *input* since it selects the derived mode of operation. A complete block diagr schematic of this ALU is shown in fig. 2.5.

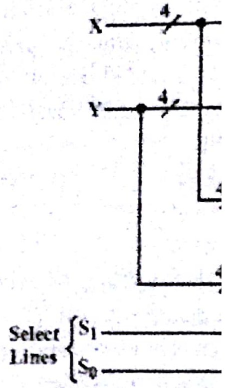


Fig. 2.5 Schematic R

The truth table illustrati

Select Lines		
S_1	S_2	
0	0	X p
0	1	X p
1	0	X ∧
1	1	X ⊕

Fig. 2.6 Truth Table Cont

Q.4. Discuss fixed poin

Ans. In the fixed-point represented as integers or fract to as fixed-point numbers be location of the decimal point assumed to be at the extreme

If the binary or decimal p then all numbers are positive or negative integers. If the radix point is assum to be at the extreme left, then all numbers are positive or negative fractions.

Consider that you have to multiply 23.15 and 33.45. This will represented as 2315 × 3345. The result will be 7743675. The decimal poi has to be placed by the user to get the correct result, which is 774.3675. So

the fixed-point representation system, the user has to keep track of the radix point, which can be a tedious job.

Q.5. What do you understand by integer representation ?

numbers can be represented d the period, or radix point. As equivalent to decimal number – it of minus sign and periods for Only binary digits 0 and 1 are n is straightforward if we are ould be used to represent the

tion in brief.

nt negative as well as positive st significant (left-most) bit in ve if the left most bit is 0 and Sign-magnitude representation that uses a sign bit. The right integer in an n bit word. For

limitations. First, addition and signs of the numbers and their peration. Second, there are two sign-magnitude representation

presentation ?

– One's complement is specific ach 1 is replaced by 0 and each one's complement of the first of each other. If one of these numbers is positive then the other number will be negative with the same magnitude and vice versa. For example $(01101)_2$ represents $(+13)_{10}$ whereas $(10010)_2$ represents $(-13)_{10}$ in this representation.

This method is widely used for representing signed numbers. In this representation also, MSB is 0 for positive numbers and 1 for negative numbers.

(ii) 2's Complement Representation – Two's complement of a bi number can be calculated by adding 1 to one's complement of the bin number. For example, 2's complement of 0101 is 1011. Since 0101 repres $(+5)_{10}$, therefore 1011 represents $(-5)_{10}$ in 2's complement representati

In this representation, if the MSB is 0 the number is positive wherea MSB is 1 the number is ne that, the 2's complement o

Q.8. What are the diff what is the best way of rep

Ans. There are various

Integer Representatic

Sign-magnitude Repr

1's and 2's Complem

The best way of represe notation is not very simple single notation for zero, wh to test for a 0 result. Also, i arithmetic addition operatio

Hence, 2's complemen numbers inside a computer.

Q.9. What is the differe of a number and the two's c

Ans. 2's complement c and the first 1 unchanged an other higher significant bits. is obtained by leaving the tw replacing 1's by 0's and 0's signed-2's complement repres of the positive number, incluc

Q.10. Explain 2's comple

(R.G.P.V., June 20

Ans. The addition of 2's complement of a number is equivalent to th subtraction of the number. Suppose, we want to subtract 0010 (2 decima from 0101 (5 decimal). If the 2's complement of 0010 is added to 0101, th sum will be 0011 (3 decimal). It is equal to 0101 (5 decimal) – 0010 (2 decima = 0011 (3 decimal).

NUMERICAL PROBLEMS

Convert hexadecimal number $(F3)_{16}$ into decimal number.

(R.G.P.V., June 2014)

5^0

$_{10}$

with the indicated bases to

(R.G.P.V., June 2015)

$1 \times 3^2 + 2 \times 3^1 + 1 \times 3^0$

$1 = (151)_{10}$ **Ans.**

$1 \times 5^1 + 0 \times 5^0$

$5 + 0$

Ans.

ng 2's complement notations eir values in decimal numbers.

(R.G.P.V., Dec. 2017)

ve. Since 01110000 represents 000 is 10010000 representing

ve. Since 11001111 represents 111 is 00110001 representing

ve. Since 10001111 represents 111 is 01110001 representing $(+113)_{10}$.

(iv) 01010101

If the LSB bit is 0, then number is positive. Since 01010101 represents $(+85)_{10}$, therefore 2's complement of 01010101 is 10101011 representing $(-85)_{10}$.

INTEGER ARITHMETIC – NEGATION, ADDITION AND SUBTRACTION, MULTIPLICATION, DIVISION

Q.11. Define arithmetic p

Ans. In a processor unit, th
called ***arithmetic processor.*** T
registers during the execution
definition of the instruction. A
decimal data, and in both case,
form. Integers or fractions m
Negative numbers may be repr
complement representation. If
included, then arithmetic proce
arithmetic operations for binar
point representation, it would b

Q.12. How do you perfor

Ans. The negation of an int
and carried out by inverting th
complement can be carried out

(i) Take the comple
sign bit.

(ii) Treating the resu

Q.13. Give the flowchart f
complement data. Explain the

Ans. The addition of two nu
of adding the numbers with the
discarded. In subtraction, first ta
add it to the minuend. When tw
sum occupies n + 1 digits, an ov
The overflow can be detected
the last two carries out of the ad

The hardware implementa
in fig. 2.7. The leftmost bits in
represent the sign bits of the n
two sign bits are added or subtracted together with the other bits in the complementer and parallel adder. If there is an overflow, then overflow flip-flop V is set to 1 and output carry is discarded.

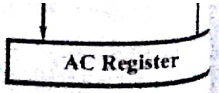
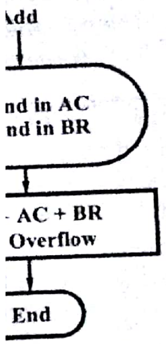


Fig. 2.7 Hardware for Signed 2's Complement Addition and Subtraction

Fig. 2.8 shows the flowchart for the algorithm of adding and subtracting two binary numbers in signed-2's complement representation. The sum is obtained by adding the AC and BR registers. If the exclusive-OR of the last carries is 1, then overflow bit V is set to 1, otherwise it is cleared to 0. For
ment of BR. An overflow
vo numbers added could



nd Subtracting Numbers
sentation

ınd subtraction for fixed
.G.P.V., Dec. 2004, 2006)

addition and subtraction
., Dec. 2007, June 2013)

addition and subtraction of
resentation with subtraction
ow one stage of the adder-
(R.G.P.V., June 2008)

n and subtraction of two
ntation. Indicate how an
(R.G.P.V., Dec. 2008)

d subtraction with signed-magnitude data. Also draw the flowchart. (R.G.P.V., June 2011)

Or

Draw flowchart to explain how addition and subtraction of two fixed point numbers can be done. Also draw a circuit using full address for the same. (R.G.P.V., Dec. 2013)

Ans. The signed-magnitude numbers are familiar because they are u in everyday arithmetic calculations. If we add or subtract two numbers A B, then we find that there are eight different conditions to consider, depend upon the sign of numbers, and the operation performed. These conditions listed in first column of table 2.1.

Table 2.1 Addition and Subt

Operation	*Add Magnitudes*
$(+A)+(+B)$	$+(A+B)$
$(+A)+(-B)$	
$(-A)+(+B)$	
$(-A)+(-B)$	$-(A+B)$
$(+A)-(+B)$	
$(+A)-(-B)$	$+(A+B)$
$(-A)-(+B)$	$-(A+B)$
$(-A)-(-B)$	

The algorithms for addition,
stated as follows –

Addition Algorithm – Whe
two magnitudes and attach the s
different, then first compare ma
from the larger, choose the sign c
complement of the sign of A if A
from A and make the sign of the

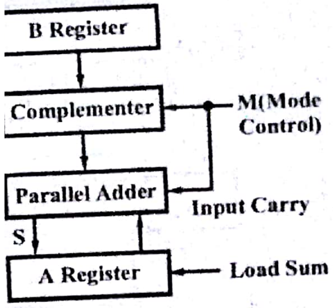
Subtraction Algorithm – W
the two magnitudes and attach the
B are identical, compare the magn
the larger. Choose the sign of the
a larger magnitude than other. If
from A and make the sign of the r

The two algorithms are similar except for the sign comparison. The procedure to be used for identical signs in the addition algorithm is the same as for different signs in the subtraction algorithm and vice versa.

Hardware Implementation – It is necessary for implementing the two arithmetic operations with hardware that the two numbers be stored in registers. Now, let A and B be the two registers that hold the magnitudes of

the numbers, and A_s and B_s be the two flip-flops that hold the corresponding signs. The result of the operation may be transferred to a third register. A saving is obtained if the result is transferred into A and A_s. So, A and A_s form
ulator register.

are for implementing the
gisters A and B and sign
to the 2's complement of
here it can be checked to



re for Signed Magnitude
and Subtraction

rry of the adder. When M =
e input carry is 0, and the
M = 1, the 1's complement
, and output $S = A+\overline{B}+1$.
mplement of $B = A-B$).

flowchart for the hardware
l by an exclusive-OR gate. If
ent; if it is 0, the signs are
s dictate that the magnitudes
tate the magnitudes be added.
e magnitudes, where EA is a
y in E constitutes an overflow
the AVF (add-overflow flip-flop).

If the signs are different for an *add* operation or identical for *subtract* operation, the two magnitudes are subtracted. The magnitudes are subtracted by adding A to the 2's complement of B. No overflow will take place if AVF is cleared to 0. A 1 in E indicates that A ≥ B and the number in A is the correct

result. Sign A_s must be made positive to avoid a negative zero, if the resul
zero. A 0 in E indicates that A < B. In this case, it is necessary to take the
complement of the value in A. This operation can be done with microoperat
$A \leftarrow \overline{A} + 1$. The 2's complement is obtained from these two microoperat
When A < B, the sign of the resu
Then, it is necessary to comple
result is obtained in register A ar
overflow indication. The final v

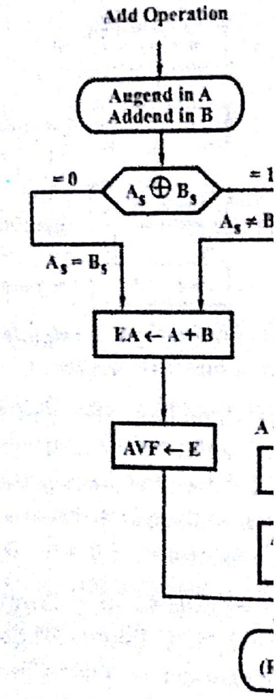


Fig. 2.10 Flowchart for

Q.15. Explain sign-magni** **their range for eight bit registe

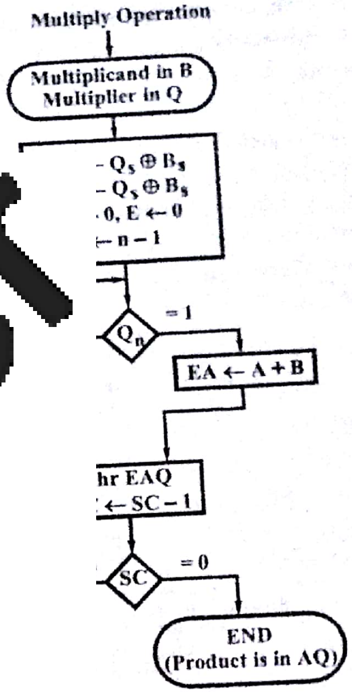
Ans. Refer to Q.6, Q.7 and Q.14.

Q.16. Explain hardware algorithm for multiplication.

(R.G.P.V., June 2003)

Ans. Algorithm for multiplication of two fixed-point binary numbers is shown in fig. 2.11. Initially, the multiplier is in Q and multiplicand in B, and

eir signs are in Q_s and B_s, respectively.
he signs are compared, and both A and Q
re set to correspond to the sign of product.
length product will be stored



1 Flowchart of Multiply Operation

ed-point binary numbers and give the hardware

ers in signed-magnitude is
e shift and add operations.

iplicand
iplier

oduct

In the example, if the multiplier bit is a 1, the multiplicand is copied down, otherwise zeros are copied down. The numbers copied down in successive lines are shifted one position to the left from previous number. At last, numbers are added and their sum forms the product. If both the numbers have the same sign, the sign of product is positive, else it is negative.

In the above process of multiplication, some changes are made implementing with a digital computer. First, instead of providing regis store and add simultaneously as many binary numbers as there are bits multiplier. It is convenient to provide an adder for the summation binary numbers and successively accumulate the partial products in

Second, instead of shifting t
shifted to the right, which se
required relative positions. T
is 0, there is no need to add al
its values.

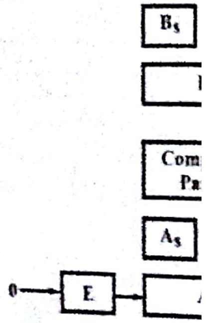


Fig. 2.12 H

Fig. 2.12 shows the hard
multiplier is stored in Q regis
initially set to a number equ
counter is decremented by
content of the counter reache
the multiplicand and Q cont
product which is moved to
shifted to the right. This shi
least significant bit of A is sl
bit from E is shifted into the r
E. After the shift, one bit of
multiplier bits one position t

Q.18. Explain Booth's
numbers. Illustrates the san
of your choice.

June 2007, Dec. 20

Or

Describe in detail Booth's multiplication algorithm and its hardw implementation.

Or ***(R.G.P.V., June 201***

Explain Booth's algorithm with in theoretical basis. (R.G.P.V., Dec. 201

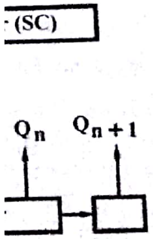
Ans. Booth algorithm provides a method for multiplying binary integers signed 2's complement representation. It operates on the fact that strings of 's in the multiplier require no addition but just shifting, and a string of 1's in multiplier from bit weight 2^k to weight 2^m can be treated as $2^{k+1} - 2^m$.

multiplier bits and shifting of
be added to the partial
naltered according to

n the multiplier bit is

partial product upon
's in the multiplier.
duct upon encountering
g of 0's in the multiplier.
tive multipliers in 2's
multiplier ends with a
tion of the appropriate

m requires the register



ion Algorithm

thm for multiplication of
appended bit Q_{n+1} are
in the multiplier (n). The
ed and if they are equal to
n encountered. If the two
he string of 1's has been
encountered. First case requires the addition of the multiplicand to the partial product in AC and the latter requires a subtraction of the multiplicand from the partial product in AC. When two bits are equal the partial product does not change. Since the addition and subtraction of the multiplicand follow each other, overflow cannot occur. The next step is to shift right the partial product

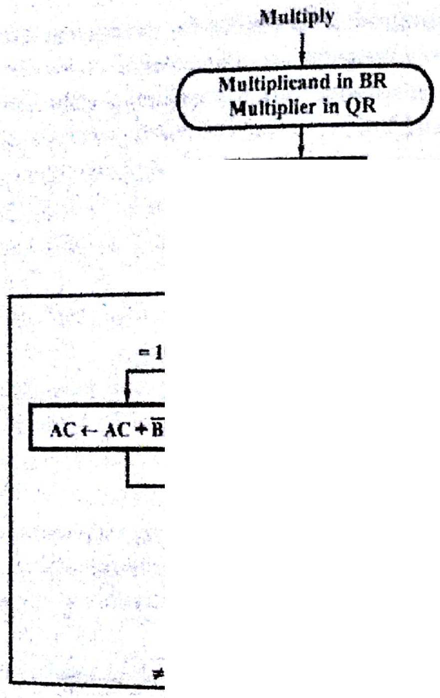


Fig. 2.14 Flowcl

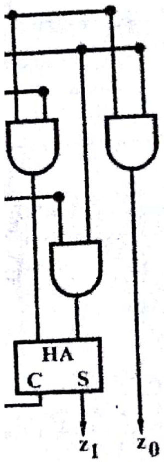
and the multiplier. This is
shifts AC and QR to the rig
decremented and loop is rep
of Booth algorithm for n =
× (– 13) = + 117.

Q_n	Q_{n+1}	BI $\overline{BR}$ +				
1	0	I S				
		as				
1	1	a				
0	1	A				
			11001			
		ashr	11100	10110	0	010
0	0	ashr	11110	01011	0	001
1	0	Subtract BR	01001			
			00111			
		ashr	00011	10101	1	000

Fig. 2.15

Q.19. Draw and explain 2-bit by 2-bit array multiplier. *(R.G.P.V., June 2011)*

Ans. Checking the bits of the multiplier one at a time and forming partial
quires a sequence of add and shift microoperations. The
one microoperation by
l at once. This is a fast
s the time only for the
multiplication array.



ltiplier

ers as shown in fig. 2.16.
its are y_1 and y_0, and the
y multiplying y_0 by x_1x_0.
luces a 1 if both y_0 and x_0
an AND operation, thus it
rtial product is formed by
y multiplying y_1, by x_1 x_0
partial products are added
there are more bits in the
dders to produce the sum.
t does not go through an
ND gate.

ore bits can be constructed.

A bit of multiplier is ANDed with all bit of multiplicand in as many levels as there are bits in the multiplier. Output in each level of AND gates is added in parallel with the partial product of the previous level to form a new partial product. At the last level AND gates produce the product. For j multiplier bits and k multiplicand bits, we require j × k AND gates and (j – 1) k-bit address to produce the product of j + k bits.

Q.20. Design an array multiplier that multiplies two 4-bit num
Use AND gates and binary adders. ***(R.G.P.V., June***

Or

Draw and explain the circuit diagram of 4-bit by 3-bit array multi

Ans. We consider a mult
four bits with a number of th
$b_3b_2b_1b_0$ and the multiplier b
gates and two 4-bit adders
shows the logic diagram of th

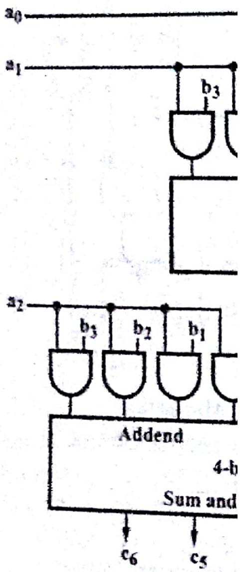


Fig. 2.17 4

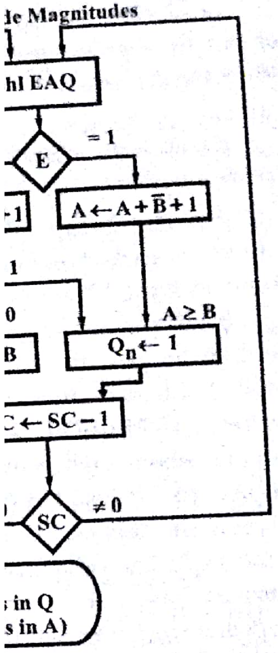
Q.21. Draw the flowcha
binary numbers in sign-mag

Draw and explain the fl

Ans. Fig. 2.18 shows the flowchart of divide algorithm. Initially, the divis
is in B and dividend in A and Q. The sign of result is transferred into Q_s to
part of quotient. SC is set with the constant which is the number of bits in th
quotient. Since an operand must be stored with its sign, one bit of the wor
will be occupied by the sign and the magnitude will consist of n – 1 bits.
divide overflow condition is tested by subtracting the divisor in B from half

ne bits of the dividend stored in A. If A ≥ B, then divide overflow flip-flop
VF is set and the operation is terminated prematurely. If A < B, no divide
verflow occurs so the value of dividend is restored by adding B to A.

Divide Operation



Operation

fting the dividend in AQ to
e shifted bit into E is 1, we
owed by n – 1 bits while B
ted from EA and 1 inserted
A is missing the high-order
his value, the 2's complement
of B results in –

$$(EA - 2^{n-1}) + (2^{n-1} - B) = EA - B$$

If we want E to remain a 1, then the carry from this addition is not transferred to E. If the shift-left operation inserts a 0 into E, the divisor is

subtracted by adding its 2's complement value and the carry is transf
into E. If E = 1, it means that A ≥ B ; therefore Q_n is set to 1. If E =
indicates that A < B and the original number is restored by adding B to A.
latter case, we leave a 0 in Q_n. This process is repeated again with regis
holding the partial remainder.
in register Q and the remain

Quotient sign is in Q_s a
original sign of the dividend.

Q.22. Explain the necess
of two fixed point binary nu
show the divide overflow co

Explain the algorithm
divide overflow ?

***Ans.* Hardware** – The h
is identical to that required f
shown in fig. 2.12. Register I
Q_n and the previous value of E
the division process. The di
length dividend is stored in re
and the divisor is subtracted by
about the relative magnitude
quotient bit 1 is inserted into
to repeat the process. If E =
remains a 0. The value of B i
to its previous value. The part
is repeated again until all five
partial remainder is shifted le
shifts, the quotient is in Q an

Divisor:
B = 10001

11
)0111000
01110
011100
-10001
-010110
--10001
--001010
---010100
----10001
----000110
-----00110

< B; enter 0 in Q; shift right B
Remainder ≥ B
Shift right B and subtract; enter 1 in Q
Remainder < B; enter 0 in Q
Final remainder

Fig. 2.19 Example of Binary Division with Digital Hardware

Flowchart for Division Algorithm – Refer to Q.21.

Divide Overflow – When the dividend is twice as long as the divisor, the
ondition for overflow can be stated as follows – A divide-overflow condition
ccurs if the high-order half bits of the dividend constitute a number greater
han or equal to the divisor. Another problem associated with division is the
must be avoided. The divide-overflow condition
se any dividend will be
ro. Overflow condition
a divide-overflow flip-

Q and SC during the
111 (multiplicand) and

ation.

A	*Q*	*SC*
000	10101	101
111		
111		
1111	11010	100
0111	11101	011
1111		
0110		
0011	01110	010
1001	10111	001
1111		
1000		
0100	01011	000

g Booth's multiplication.

Sol. The step-by-step multiplication process using Booth's algorithm for the given numbers is shown in table 2.3. Here,

BR = 11001 (–7) {2's complement}

$\overline{BR} + 1 = 00111$

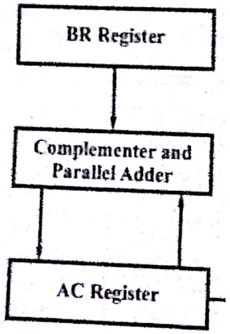
QR = 00011

Table 2.3

Q_n	Q_{n+1}	$BR = 11001$ $\overline{BR} + 1 = 00111$	AC	QR	Q_{n+1}	Sc
		Initia	00000	00011	0	
1	0	Subtrac				
		ash				
1	1	ash				
0	1	Add				
		ash				
0	0	ash				
0	0	ash				

Prob.6. Draw the req ***multiplication using Booths*** ***(–6) and 4.***

Sol. The required registe Booth algorithm is given belo



Multiplication of (–6) an using Booth algorithm is given in tab 2.4.

Here, $BR = 11010$ (–6) {in 2's complement}

$\overline{BR} + 1 = 00110$

$QR = 00100$ (+4)

Table 2.4 Multiplication of (–6) * (4) with Booth Algorithm

Q_n	Q_{n+1}	$BR = 11010$ $\overline{BR} + 1 = 00110$	AC	QR	Q_{n+1}	SC
		Initial	00000	00100	0	101
					0	100
					0	011
					1	010
					0	001
					0	000

n process using Booth *are multiplied. Assume*

. ***(R.G.P.V., June 2009)***

sing Booth algorithm for

11)$_2$

				QR	Q_{n+1}	SC
				01101	0	101
				10110	1	100
				11011	0	011
		ashr	11010	01101	1	010
1	1	ashr	11101	00110	1	001
0	1	Add BR	01111 01100			
		ashr	00110	00011	0	000
			+ 195			

(ii) (+ 15) × (– 13) = –195 = (1100111101)2's complement

BR = 01111 (+ 15)

QR = 10011 (– 13)

Table 2.6

Q_n	Q_{n+1}	$BR = 01111$ $\overline{BR} + 1 = 1$
		Initial
1	0	Subtract BF
		ashr
1	1	ashr
0	1	Add BR
		ashr
0	0	ashr
1	0	Subtract BR
		ashr

Prob.8. Explain Booth's
step multiplication process usi
(+15) and (–13) in binary.

Sol. **Booth's Multiplicati**

Also refer to Prob.7 (ii).

Prob.9. Explain Booth's
table for register contents usec
multiplier = – 6 and multiplica

Sol. **Booth's Algorithm –**

Now, multiplication of 5 *
table 2.7. It shows the step-by-s
the multiplier in QR is negative
10-bit product appears in AC anc
is the original sign bit of the mult

Here,

$BR = 00101\ (+5)$

$\overline{BR} + 1 = 11011\ (-5)$

$QR = 11010\ (-6)$ {2's complement}

Table 2.7 Multiplication of 5 * (– 6) with Booth Algorithm

Q_n	Q_{n+1}	$BR = 00101$ $\overline{BR} + 1 = 11011$	AC	QR	Q_{n+1}	SC
		Initially	00000	11010	0	101
		ashr	00000	01101	0	100
				0	1	011
				11	0	010
				01	1	001
				10	1	000

using Booth method.
(R.G.P.V., May 2018)

er to Q.18.

0 1 0
1 0 –1
1 1 0
0 0 ×
0 × ×
× × ×
1 1 0 (–10)

how how to perform the
(R.G.P.V., June 2017)

ollowing here –

– 16 into 18 + (– 16).

nent

–

0 0 0 1 0 0 1 0 → 18
\+ 1 1 1 1 0 0 0 0 → 2's Complement of 16

Discarded Bits → ① 0 0 0 0 0 0 1 0

= 2

Ans.

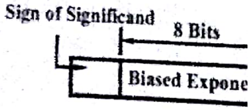
FLOATING-POINT REPRESENTATION, FLOATING-POIN ARITHMETIC

Q.23. Explain floating-point representation with example.

Ans. With a fixed-point n
and negative integers centere
this format allows the represe
well. But this approach has
represented, nor can very sma
in a division could be lost. I
limitation. Thus 456,000,00(
fraction 0.000000000456 ca
we only slide the decimal poi
10 to keep tracking of that deci
have a facility for specifying
the floating-point number is
point numbers which are spe

A floating-point numbe
significand (also called mant
a number obtained from mul
thus –

The base B is implicit a
numbers. Fig. 2.21 shows a t
of representing a binary floa



$0.11010001 \times 2^{1010}$

$-0.11010001 \times 2^{101}$

$0.011010001 \times 2^{-1010}$

$-0.11010001 \times 2^{-10100} = 1\ \ 01101011\ \ 10100010000000000000000$

(b) Examples

Fig. 2.21 32-bit Floating-point Format

Computers with shorter word length use two or more words to repres
a floating-point number.

Q.24. What is the parity-bit and why we use it ? (R.G.P.V., Dec. 2015)

Ans. An additional bit called a parity bit is added to each data word. The
dditional bit is so chosen that the weight of the code word so formed is either
ven (even parity) or odd (odd parity). When a single error or an odd number
the parity of the code word changes.
id and violation of the
de word.

o we do normalization

said to be normalized.
alized numbers because
are many unnormalized

most significant digit of
ition of the radix point is
lized only if its leftmost
110 in binary form is not
the number two positions
011000, the number can
ough 4 (2^2). The exponent
or floating-point number.
de the maximum posssible
does not have a nonzero
ll 0's in the mantissa and

ific computations because
computations. Although,
s are more difficult than
heir execution takes longer

oint arithmetic ? Explain.

point numbers are more
eir execution takes longer
and requires more complex hardware. In addition or subtraction, it is necessary to ensure that both operands have the same exponent value. Thus, an alignment of the radix points is required before adding or subtracting the mantissas. Table 2.8 summarizes the basic operations for floating-point arithmetic.

Table 2.8 Floating-point Arithmetic Operations

Floating-point Numbers	Arithmetic Operations
$X = X_s \times B^{X_E}$ $Y = Y_s \times B^{Y_E}$	$X+Y = \left(X_s \times B^{X_E - Y_E} + Y_s\right) \times B^{Y_E}$

Examples –

$X = 0.$

$Y = 0.$

$X+Y = (0$

$X-Y = (0$

$X \times Y = (0$

$X \div Y = (0$

Considering the sum of

+

In above example, it is n
mantissas can be added. We
left or shift the second numbe
stored in registers, shifting to
Shifting to the right causes a lo
an error while the second me
method is preferable. Multipl

Problems arised as the r

(i) Significand U
digits flow off the right end
is required.

(ii) Significand O
same sign may result in a ca
fixed by realignment.

(iii) Exponent Overflow – A positive exponent exceeds the maxim
possible exponent value. In some system, this may be designated as + ∞ or

(iv) Exponent Underflow – A negative exponent means it is less t
the minimum possible exponent value. This means that the number is
small to be represented and it may be reported as 0.

Q.27. Explain in short with the help of flowchart, how the addition and subtraction is carried out of floating-point numbers. ***(R.G.P.V., June 2004)***

Or

Draw and explain flowchart for addition and subtraction of floating-point numbers. ***(R.G.P.V., June 2005, 2012)***

ldition and subtraction –
gn the mantissas
malize the result.

addition or subtraction
gisters that will be used
pt for a sign change, so
end, if it is a subtract
reported as result.

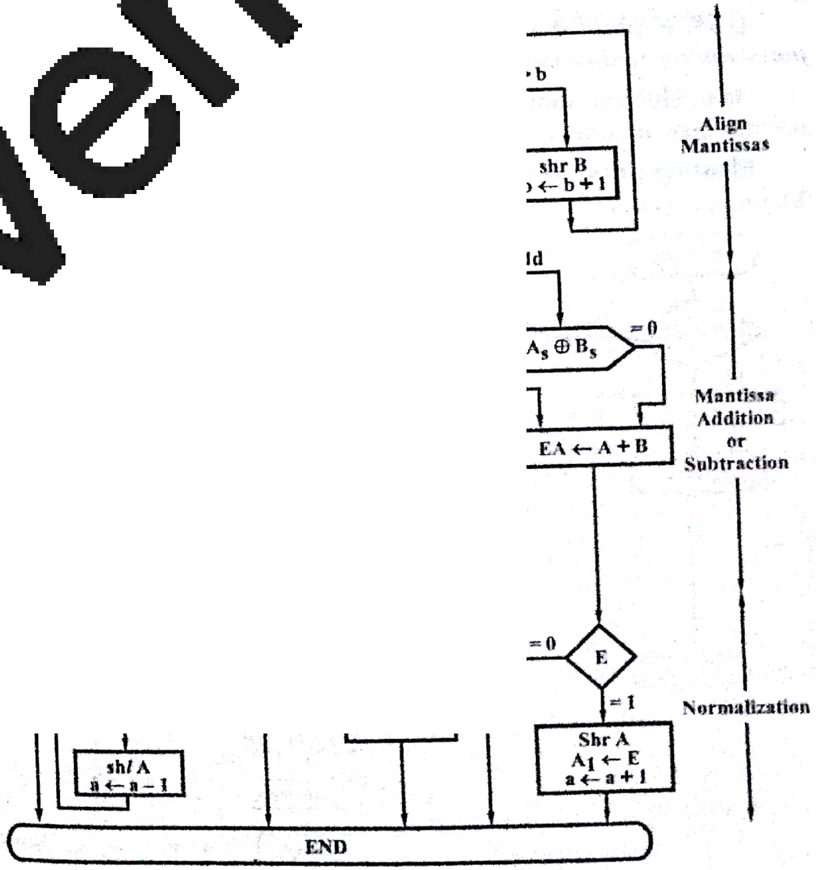


Fig. 2.22 Addition and Subtraction of Floating-point Numbers

Next phase is to manipulate the numbers so that the two exponents equal. Alignment may be achieved by shifting either the smaller number to right or shifting the larger number to the left. Because either operation result in the loss of digits, it is the smaller number that is shifted; any d that are lost are therefore of relatively small significance. Alignment is achi

by shifting the magnitude port

the exponent until the two ex

zero for the significand, then

Next the two significand

signs. If signs differ, the result n

overflow by 1 digit. If so, the

exponent is incremented. The

could be reported and the ope

Normalization consists of shift

a decrement of exponent and

the result may be rounded of

Q.28. Explain the multi

point numbers with their res

Ans. Multiplication and

simpler than addition and sub

Floating-point Multipli

floating-point numbers is sho

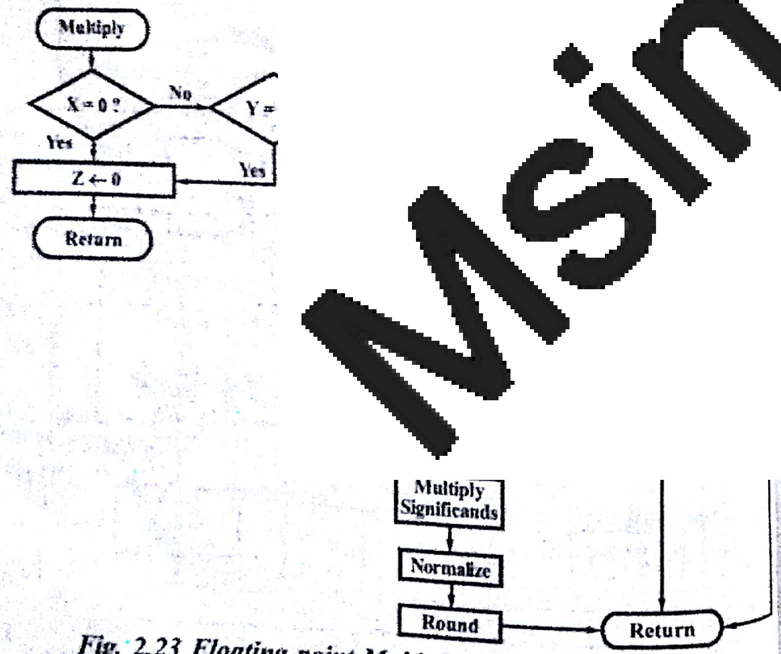


Fig. 2.23 Floating-point Multiplication (Z ← X × Y)

The multiplication algorithm can be subdivided into four parts –

(i) Check for zeros (ii) Add the exponents

(iii) Multiply the mantissas (iv) Normalize the product.

if either operand is 0, 0 is reported as the result. The next step

form, the exponent sum

must be subtracted from

exponent overflow or

rithm.

er range, the next step is

formed in the same way

ing with a sign magnitude

gth of the multiplier and

the product is calculated,

division is depicted in

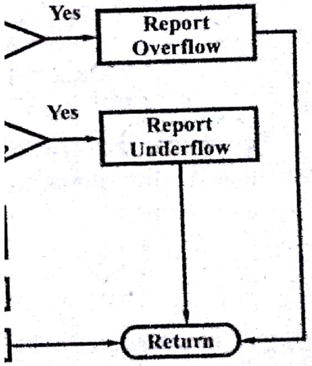


Fig. 2.24 Floating-point Division (Z ← X/Y)

The algorithm is subdivided into five parts –

(i) Check for zeros. (ii) Initialize registers and evaluate the sign.

(iii) Align the dividend. (iv) Subtract the exponents.

(v) Divide the mantissas.

Again, the first step is testing for 0. If the divisor is zero, an error rep is issued, or the result is set to infinity. A dividend of 0, results in 0. In second step, the divisor exponent is subtracted from the dividend expon This removes the bias which must be added back in. After that tests are m for exponent underflow or overflow. The magnitudes of the mantissa divided as in fixed-point case.

Q.29. How is multiplicati
Explain using flow chart.

Ans. Refer to Q.28.

NUMER

Prob.12. Represent the nu number with 24 bits. The norm exponent has 8 bits.

Sol. The binary equivalent o

```
2 | 46  0
2 | 23  1
2 | 11  1
2 |  5  1
2 |  2  0
2 |  1  1
  |  ×
```

46 = 1 0 1 1

$.5 \times 2 = 1$

i.e., $(46.5)_{10} = (1\ 0\ 1$

The normalized fractions m bits, then floating-point binary n

Sign

↓

0 1 0 1 1 1 0 1 0 0

16-bit mant

HARDWIRED, MICRO CONTROL UNIT, CONTROL MEMORY, MICROPROGRAM SEQUENCE

Q.30. What is meant by hardwired control ? ***(R.G.P.V., Dec. 201***

Ans. When the control signals are generated by hardware using convention logic design techniques, the control unit is said to be ***hardwired.***

Q.31. Explain the hardwired control unit in detail. (R.G.P.V., Dec. 2011)

Or

Write short note on hardwired control unit. (R.G.P.V., Dec. 2005, 2012)

Or

control unit. (R.G.P.V., Dec. 2015)

rol unit.

R.G.P.V., Dec. 2017)

by hardware using said to be ***hardwired.***

g. 2.25. This control a clock signal, CLK. ed by the following

status flags.

g the state of various ed to it.

ts

tion es

tion

of the control unit, we will start by giving a simplified view of the hardware involved. The decoder-encoder block in fig. 2.25 is a combinational circuit which produces the required control outputs, depending on the state of all its inputs. By separating the decoding and encoding functions, we obtain the more detailed block diagram in fig. 2.26.

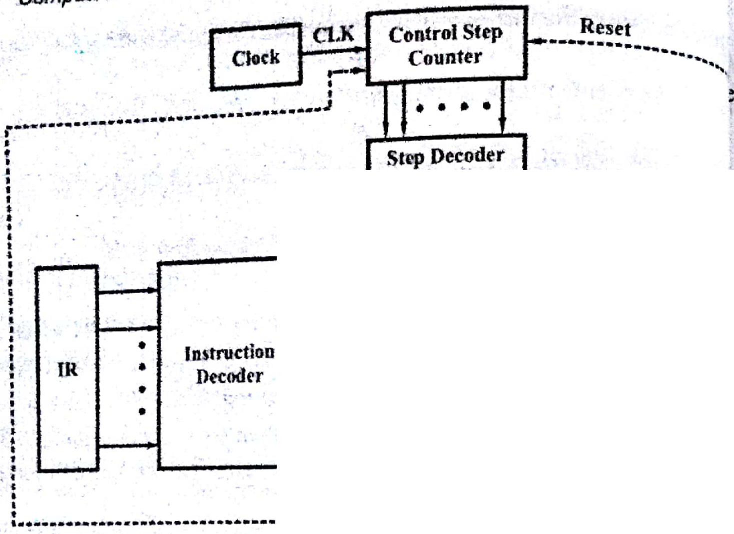


Fig. 2.26 Separation o

The step decoder provide
in the control sequence. Like
the instruction decoder has a s
machine instruction. It means th
loaded in the IR, one of the
INS_m is set to 1, and all other

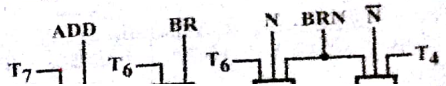
All input signals to the e
2.26 should be combined to ge
control signals Y_{in}, PC_{out}, A
Structure of the encoder is
circuit shown in fig. 2.27, that i
function –

$$Z_{in} = T_1 + T_6.ADD + T$$

It means that the control
turned on during time slot T
instruction, and so on. This p
function has been complic
the control sequences. The term T_1 is common to all instructions, becau
takes place during the fetch phase. Likewise, the End control signal, fig. 2
is generated from the logic function –

$$End = T_7.ADD + T_6.BR + (T_6.N + T_4.\overline{N}).BRN + \ldots\ldots$$

Fig. 2.28 shows how the End signal can be used to start a new instruction
etch cycle by resetting the control step counter to its starting value.



Signal

ned control unit with
R.G.P.V., Dec. 2008)

plain the working of
R.G.P.V., June 2009)

e organization of a
(R.G.P.V., Dec. 2011)

ol unit block diagram.
(R.G.P.V., Dec. 2012)
ammed control unit is

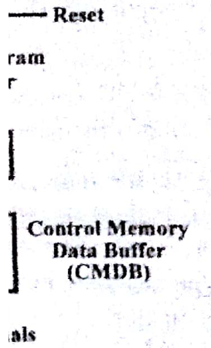


Fig. 2.29 Block Diagram of Microprogrammed Control Unit

A summary of the use of various components included in this organization is given below –

(i) Control Memory Buffer Register (CMBR) – Control memory buffer register functions the same as the memory buffer register of the main

memory. Basically, it is a latch, and serves as a buffer for the microinstruc
retrieved from the control memory. Typically, each microinstruction will
three distinct fields.

Condition Select	Branch Address	Control Function

[redacted]

The *condition select fiel*
case the selected condition is
the output of the multiplexer
microprogram counter (MPC),
with the address specified in t
If the selected external condi
point to the next microinstruct
conditional branching. The co
hold the control information i

(ii) Microprogram
(MPC) holds the address of th
it is loaded from an external
microprogram to be executed.
is incremented after each instr
to the control memory buffe
encountered, then the microp
contents of the branch addres
control memory buffer regist

(iii) External Cond
the external conditions accor
of the microinstruction. So, th
encoded form. Any encoding
small control memory; thus,

Q.33. What do you unde
the block diagram of microp

Ans. Refer to Q.38 and

Q.34. Discuss in brief microprogram control unit and hardwired co
unit.

(R.G.P.V., June 2005, 2

Or

Explain hardwired, microprogrammed control unit.

(R.G.P.V., June 2

Ans. Refer to Q.32 and Q.31.

Q.35. Explain various branching techniques used in microprogrammed
ontrol unit.

(R.G.P.V., June 2007)

Ans. A variety of approaches have been taken for dealing with conditional

uffers a penalty for a
instructions to fetch
proach is to replicate
fetch both instructions,
ith this approach –
ention delays for access

nter the pipeline before
tion needs an additional

conditional branch is
dition to the instruction
he branch instruction is
y been prefetched.
ery-high-speed memory
ne and containing the n
ranch is to be taken, the
thin the buffer. If so, the

es can be used to predict
mon are the following –

depend on the execution
struction. The latter two
tion history.
se either always assume
that the branch will not be taken and continue to fetch instructions in sequence,
or they always assume that the branch will be taken and always fetch from the
branch target.

The final static approach makes the decision based on the opcode of the branch instruction. The processor assumes that the branch will be taken for certain branch opcodes and not for others.

Dynamic branch strategies attempt to improve the accuracy of pred
by recording the history of conditional branch instructions in a program
example, one or more bits can be associated with each conditional br
instruction that reflect the recent history of the instruction. These bit
referred to as a taken/not taken switch that directs the processor to
particular decision the next tir

(v) Delayed Branch
by automatically rearranging i
instructions occur later than ac

Q.36. Explain the differ
programmed control. Is it po
with a control memory ? Writ
(R.G

Differentiate between ha
control unit.

Compare hardwired and
relative merits and demerits.

Hardwired control unit is
Justify this statement.

Differentiate hardwired a
merits and demerits of each

***Ans.* Difference between**
Control – There are two majo

(i) Hardwired control

Hardwired control unit is fa
the use of combinational circuit
is more advantageous as comp
organization, the control logic
and other digital circuits. It h
produce a fast mode of operati

Mostly computer based on reduced instruction set computer (RI
architecture concept use hardwired control, as the name implies, requ
changes in the wiring among the various components, if the design has to
modified or changed.

In the microprogrammed organization, the control information is stored
a control memory. The control memory is programmed to initiate the requi

equence of microoperation. The main advantage of microprogrammed control
s the fact that once the hardwired configuration is established, there should
e no need for further hardware or wiring changes.

Is it Possible to have a Hardwired Control Associated with a Control
Hardwired control, by definition, does not contain a control memory.

grammed Control –
nicroprogramming to
of the control unit.
mplement.
n complex logic for
nstruction cycle.
f a microprogrammed

nmed Control –
rogrammed unit that it
nparable technology.
nique for implementing
implementation.
nction format, typically

ter detail.

n computer ? What is
nd disadvantages ?
(R.G.P.V., Dec. 2010)

The control unit is used
ata items to the selected
bility of a control unit is
nerating a set of signals
chronization establishes

36.

ogrammed control unit?
(R.G.P.V., Dec. 2014)

Ans. Microprogramming is a method of control unit design in which the control signal selection and sequencing information is stored in a ROM or RAM called a control memory. The control signals to be activated at any time are specified by a microinstruction, that is fetched from control memory in much the same way an instruction is fetched from main memory. In addition,

each microinstruction explicitly or implicitly specifies the next microinstr to be used, thereby providing the necessary information for microoper sequencing. A set of related microinstructions forms a microprog Microprograms can be altered relatively easily by altering the contents of c memory; thus microprogramming yields control units which are more n than their hardwired counte

hardware cost owing to the

also a performance penal

microinstructions from contr

the use of microprogrammi

where chip area and circuit d

continues to be used in such

A control unit whose b

called a microprogrammed c

Q.39. Draw the funct

computer.

Ans. Fig. 2.30 shows th

consists of two decoders – a

gates. An instruction read f

(IR). The instruction register

– the I bit, the operation cod

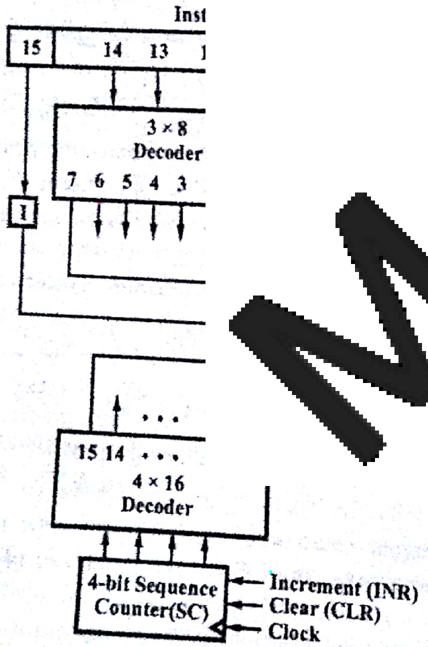


Fig. 2.30 Functional Block Diagram of Control Unit

decoder are represented by the symbols D_0 through D_7. The subscripted decimal number is equivalent to the binary value of the corresponding operation code. Bit 15 of the instruction is shifted to a flip-flop denoted by the symbol I. Bits 0 through 11 are applied to the control logic gates. The 4-bit sequence

from 0 through 15. The outputs of the counter are

or cleared synchronously.

ide the sequence of timing

e counter is cleared to 0.

V., Dec. 2012, May 2018)

8, Dec. 2008, June 2011)

is referred to as a *control*

n of the complete operation

operations to be performed

of these cycles.

(R.G.P.V., Dec. 2007)

ent microinstruction while

mory. The data register is

ster. (R.G.P.V., June 2011)

pecifies the address of the

ls the microinstruction read

trol word that specifies one

c. 2008, 2009, June 2012)

V., Dec. 2007, June 2011)

vithin it a *microinstruction.*

operations for the system.

Q.44. Describe a typical microinstruction format. What are the considerations in the design of a microinstruction format ? ***(R.G.P.V., June 2002)***

Or

Write short note on microinstruction format.

(R.G.P.V., Dec. 2006, June 2008)

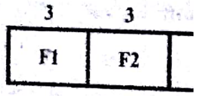
Or

What is microinstruction format ? Explain different fiel
microinstruction. **(R.G.P.V., June 2**

Or

Explain in detail various fields of microinstruction fo
diagram.

Ans. Fig. 2.31 shows th
memory. The 20 bits of the m
parts – the three fields F1,
computer; the CD field choos
the type of branch to be usec
The address field is seven
128 = 2^7 words.



F1, F2, F3 : Microoper
CD : Condition for Br
BR : Branch Field
AD : Address Field

Fig. 2.31 Mi

Microoperations are subd
each field, encoded three bits sp
are listed in table 2.9. This
microinstruction only three mi
field. If less than three microop
the binary code 000 for no ope

Each microoperation is de
transfer type microoperations
designate the source register, th
letters designate the destination

The CD field consists of tw
bit conditions which are listed i
that a reference to CD = 00 will always find the condition to be true. When
is used in conjunction with the BR field, it provides an unconditional branc
operation. The BR field consists of two bits. It is used in conjunction with th
address field AD to choose the address of the next microinstruction as show
in table 2.9.

Table 2.9 Binary Code and Symbols for Microinstruction Fields

F1	*Microoperation*	*Symbol*
000	None	NOP

F1	*Microoperation*	*Symbol*
000	None	NOP
	AC + DR	ADD
	0	CLRAC
	AC + 1	INCAC
	DR	DRTAC
	DR(0 – 10)	DRTAR
	PC	PCTAR
	] ← DR	WRITE

Symbol	*Comments*
U	Unconditional branch
I	Indirect address bit
S	Sign bit of AC
Z	Zero value in AC

n = 0
- 1 if condition = 1
n = 0
ubroutine)
, CAR(0,1,6) ← 0

Q.43. Define microprogram.
(R.G.P.V., June 2004, Dec. 2008, 2009, June 2012, 2016)

Or

Explain the term microprogram. **(R.G.P.V., Dec. 2007, June 2011)**

Ans. A sequence of microinstructions constitutes a ***microprogram.***

Q.46. *Define microcode.*

(R.G.P.V., June 2004, Dec. 2008, 2009, June

Or

Explain the term microcode. *(R.G.P.V., Dec.*

Ans. It is same as micro

Q.47. *Define the term mi*

Explain the term sequer

Ans. The next address g

sequencer, as it determines

memory. The address of the n

ways, depending on the sequ

sequencer are incrementing t

the control address register

an external address, or loadin

Q.48. *Define the term mi*

Ans. Microinstruction s

the control memory.

Q.49. *What are the mi*

sequencing ?

Ans. There are two conc

sequencing technique – the

generation time. The first co

memory decreases the cost o

desire to execute microinstr

Q.50. *What is the purp*

internal structure and work

A microprogram seque

the block diagram of the s

operations which are requ

subroutine micro instructions.

call and return f

(R.G.P.V., June 200

Or

With a neat block diagram, explain the working principle of micro-program sequencer.

(R.G.P.V., June 2009, Dec. 2010, 201

Or

Draw and explain the microprogrammed control unit with next address generation. *(R.G.P.V., June 2013)*

Or

Write a brief note on microprogram sequencer. *(R.G.P.V., June 2014)*

selection part of the

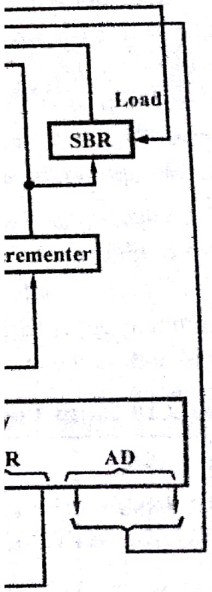
oprogram sequencer is

icroinstruction may be

cer decides the specific

egister. The selection of

formation bits that the



cer for a Control Memory

oprogram sequencer. The

w the interaction between

s circuit, two multiplexers

from one of four sources and routes it into a control address register (CAR). The second multiplexer tests the value of a chosen status bit and the result of the test is applied to an input logic circuit. The output from control address register (CAR) provides the address for the control memory. The content of control address register (CAR) is incremented and given to one of the multiplexer inputs and the

subroutine register (SBR). Other three inputs to multiplexer number 1 from the address field of the present microinstruction, from the outp subroutine register (SBR), and from an external source which map instruction. The diagram shows a single subroutine register, but a ty sequencer will have a registe stack about four to eight levels manner, a number of subroutir with a stack pointer, a push a address during the call and re

The condition (CD) field bits in the second multiplexer variable is equal to 1; otherwi two bits from the branch (BI particular sequencer, the input are available in the unit. The branch or jump, call and ret push or pop the stack, and oth can provide upto eight addre

In the input logic circui and three outputs S_0, S_1 and addresses for control address in subroutine register (SBR). decide the path in the multip

Table 2.10 Input Logic

BR	I
Field	I_1
0 0	0
0 0	0
0 1	0
0 1	0
1 0	1
1 1	1

Table 2.10 shows the tr input logic circuit. Inputs I_1 I_0 are similar to the bit values in the BR field. Bit values for S_1 and S_0 determined from the stated function and the path in the multiplexer wh establishes the needed transfer. During a call microinstruction (BR = 01), subroutine register (SBR) is loaded with the incremented value of con address register, provided that the status bit condition is fulfilled (T = 1).

With the help of truth table, the simplified Boolean functions for the input logic circuit can be obtained as –

$$S_1 = I_1$$

$$S_0 = I_1 I_0 + I_1' T$$

gates, an OR gate, and

and explain how a ***(R.G.P.V., June 2010)***

ganization. Give their ***June 2013, Dec. 2014)***

cal organizations are as

cal Organization

al organization, there is ant encoding of the nformation.

ower operating speed is t is considered to be useful.

limited ability to express microoperations.

s of vertical organization ort.

es of horizontal and vertical

(i) When operating speed of computer is an important factor and where the parallel usage of a number of resources is permitted by the machine structure, the horizontal organization approach is preferable.

(ii) The vertical organization approach is suitable when operating speed of computer is slower and less bits are needed in the microinstruction.

(iii) The considerable factor is vertical approach is the requirement for the parallel hardware needed to handle the execution of instruction.

NUMERICAL PROBLEMS

Prob.13. Show how a 9-
can be divided into subfiel
microoperations can be speci

Sol. Since a field of 5 bits
a field of 4 bits can specify 15
specify 46 microoperations.

One microinstruction can

UNIT

ORGANIZATION,
IOTATION,
TWO, THREE-

ulk of data processing
). It is responsible for
r system. It is referred
all major calculations

t/output devices.

put/output devices is
tem, data is transferred
rocessor to an output
evice controller, which
ices. Actually, the CPU
g the I/O operations.

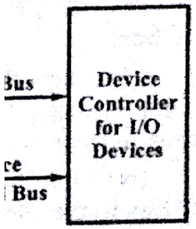


Fig. 3.1 CPU to I/O Devices Communication

In the computer system, the interface unit works as an intermediary between the processor and the device controllers of various peripheral devices. The interface unit accepts the control commands from the processor and

interprets the commands so that they can be easily understood by the d
controllers for performing the required operations. Hence, the interface u
responsible for controlling the input and output operations. The process
I/O devices communication involves two important operations – I/O read
I/O write.

The I/O operation helps th
The sequence of steps done du
to the CPU are as follows –

(i) The input device
the data bus which transfers sir

(ii) The input device tl
control bus to the data register, s

(iii) After accepting t
unit, it issues a data accepted
acknowledgement to the input d
the input device disables the dat

(iv) The flag bit of th
register holds the data.

(v) Now, the CPU iss
the interface unit.

(vi) The data register t
to the CPU. When the data is re
signal to the input device, showi

The I/O write operation hel
device. The sequence of steps d
the output device are as follows

(i) The CPU kept the
bus connected to the data registe

(ii) The CPU also kept
address bus.

(iii) The CPU then issu
on the data register. The data reg
the status register is set to 1.

(iv) Now, the data regis
control bus to the CPU, showing that the data has been received.

(v) Then, the interface unit kept the data on the data bus connect
to the device controller of the output device.

(vi) The output device receives the data and sends an acknowledg
ment signal to the CPU through the interface unit, showing that the desir
data has been recieved.

Q.3. Explain stack organization in detail.

Ans. A storage device that stores information like a manner that the item stored last is the first item is known as ***stack***. Stack is the useful feature of
stack is essentially a memory unit with an address
ster which holds the
t the top item in the
may be taken out or
ailable for writing or
deleted.

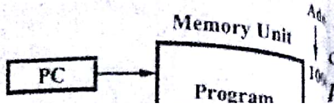
a stack. The operation
the result of removing
sertion is called push
item on top. Nothing
ions are simulated by

collection of a finite
iced in a portion of a
stack is shown in fig.
whose value is equal
he stack. Three items
em C is on top of the
popped by reading the
itent of SP to remove

w on top of the stack.
vord in the next higher
been read out but not
n the stack is pushed a

memory system of a
R.G.P.V., June 2017)

is done by assigning a portion of memory to a stack operation and using a processor register as a stack pointer. A portion of computer memory partitioned into three segments program, data and stack is shown in fig. 3.2. In the program, the program counter PC points at the address of the next instruction. The address register AR points at an array of data.



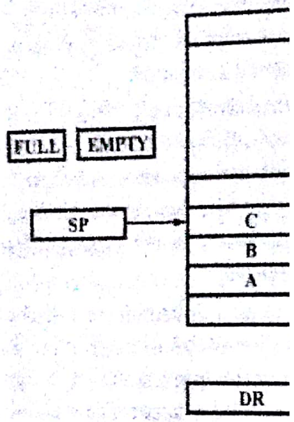


Fig. 3.2 Block Diagram of a 64

The three registers are c
one can give an address for m
from the stack. From fig. 3.3
and the stack grows with decr
the stack is at address 4000,
the last address which can be
available for stack limit check

Q.5. Discuss polish nota

Ans. The method of writi
their operands or after them is
to write an expression –

(i) Infix Notation.
operands then the expression
expression to add two number

infix notation as

A + B

(ii) Prefix Notations – When the operators are written before the operands, then the expression is called the prefix notation or prefix polis notation. For example, the expression to add two numbers A and B is writt in prefix notation as –

+AB

(iii) Postfix Notations – When the operators are written after the operands, it is called postfix or suffix notations. It is also known as ***reverse polish notation.*** For example, the expression to add two numbers A and B is in postfix notation as –

ate infix and postfix
.G.P.V., May 2018)
to store register pair

operand are copied
pointer register is
ter (B, D, H, A) are
ecremented again and
opied to that location.

Stack (i.e. SP) = 2000.

es in memory as shown

extract data from stack

ut by the stack pointer
L, status flags) of the
contents of that memory
, A) of the operand. The

Example : POP B

Suppose B contains 00 and C contains 00 and stack pointer = 1998

00 00

The POP B instruction makes the following changes in register as shown and top of stack is incremented by 2.

i.e. new top of stack = SP + 2 = 1998 + 2 = 2000

34 56

Memory Location	Data
2000	

Difference between In

Q.7. Explain how the ev
a stack.

Write an algorithm to e
structure.

Ans. Once the expressio
remember the precedence rul
from left to right. We push th
we come across an operator
pushing back the result of the
for use as an operand of the

Let us write an algorithm

```
clear the stack
symb = next input chara
while not end of input
{
 if symb is an operand
     push onto the stack
 else
 {
     pop two operands fr
     Result = op1 symbol
     push result onto the
 }
symb = next input charac
}
return(pop stack)
```

For example – Suppose, we have to evaluate the following postfi
expression –

98 + 382 / * 2 + –

If we workout our algorithm with this input, we can show the contents of the stack, symb, op1, op2 and result after each iteration of the loop as in table 3.1.

Table 3.1

S No.	Symbol	op1	op2	Result	Stack
					9
					9, 8
					17
					17, 3
					17, 3, 8
					17, 3, 8, 2
					17, 3, 4
					17, 12
					17, 12, 2
					17, 14
					3

ize of stack keeps varying
tack.

(R.G.P.V., June 2017)

wn in a rectangular box
ear in memory words or in
titioned into groups known
ormats are –

ie operation to be performed.

ory address or a processor

the operand or the effective

rts. (R.G.P.V., June 2014)

at instruct the computer to
parts. The most basic part
tion code of an instruction
, subtract, and so on.

specifies the operation to be performed. This operation must be performed on some data stored in processor registers or in memory. Therefore, an instruction code must specify not only the operation but also the registers or the memory words where the operands are to be found, as well as the register or memory word where the result is to be stored.

Q.10. What is an instruction ? What are the different parts of instruction ? **(R.G.P.V., June 2**

Ans. An instruction manipulates the stored data and a sequence of instruc constitutes a program. In general, an instruction consists of two parts –

(i) Opcode field (ii) Address field (s).

The opcode field specifie
may reside within a CPU regi
address field is to indicate the
read from or stored into two o
more than one address. For ex

Add

Opcode field

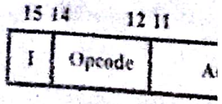
Assume that this comput
the destination register. The
of CPU registers R_0 and R_1 a
and types of instructions supp
to another and depend primari

Q.11. Write down the inst

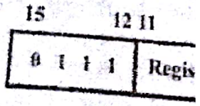
Draw the basic computer in
reference, and input-output ty

Write in brief different type

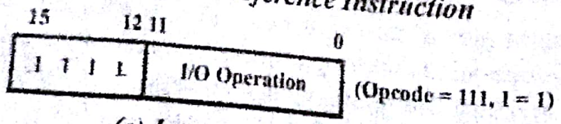
Ans. A basic computer ha
reference, register reference, a
in fig. 3.4.



(a) Memor



(b) Registe. -reference Instruction



(c) Input-output Instruction

Fig. 3.4 Instruction Formats of Computer

Each format consists of 16 bits. The operation code (opcode) part of the istruction has three bits and the mean ing of the remaining 13 bits is determined y the operation code encountered.

In a ***memory-reference*** instruction, 12 bits are used to specify an address bit to specify the addressing mode I. Addressing mode bit (I) is equal

he operation code 111 with a
e of instruction specifies an
erand from memory is not
on or test to be executed.
reference to memory and
in the leftmost bit in the
the type of input-output

table 3.2.

ructions

scription

word to AC
vord to AC
word to AC
of AC in memory
itionally
ve return address
skip if zero

C

AC and E
C and E

uction if AC positive
uction if AC negative
uction if AC zero
uction if E is zero

HLT		7001	Halt computer
INP		F800	Input character to AC
OUT		F400	Output character from AC
SKI		F200	Skip on input flag
SKO		F100	Skip on output flag
ION		F080	Interrupt on
IOF		F040	Interrupt off

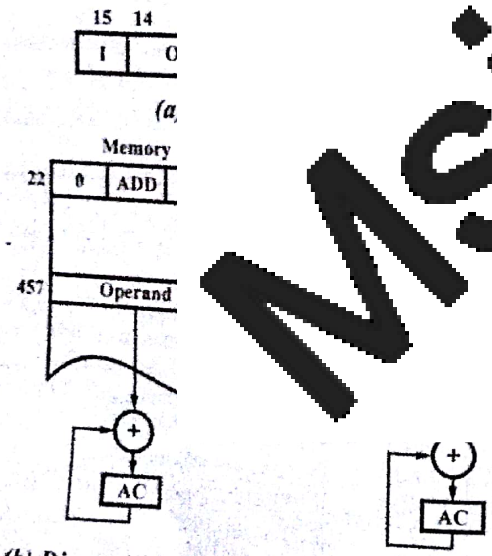
Q.12. What is instruction format ? Explain various instruction fo
(R.G.P.V., June

Ans. Refer to Q.8 and Q.11.

Q.13. What is the difference between a direct and indirect
instruction ? How many ref
instruction to bring an opera
example of each type.

Ans. In some cases, it is
code not as an address, but
instruction code specifies an
immediate operand. If the s
then it is said that the instru
possibility called indirect ad
instruction specify an addres
operand is obtained. In the
between a direct and an indi

To demonstrate this confi
depicted in fig. 3.5. This inst
12-bit address, and an indir
address, mode bit is 0, and fo
direct address instruction. It i



(b) Direct Address ***(c) Indirect Address***

Fig. 3.5 Demonstration of Direct and Indirect Address

ero, the instruction is identified as a direct address instruction. The opcode
pecifies an ADD instruction, and the address part is the binary equivalent of
57. Control finds the operand in memory at address 457 and adds it to the
ontent of AC. In fig. 3.5 (c), since the mode bit (I) of the instruction in address
an indirect address instruction. The address part is the
address 300 to find the
the operand. Then, the
of AC.

ces to memory to fetch
ess of the operand; the
ned to be the address of
get address in a branch-
ruction of fig. 3.5 (b) is

to memory –

needs three references

ress (iii) Read operand.

he accumulator in the
I.
port C.
form.
r.

0H
memory location whose

2500H
memory, whose address
tent of the accumulator.

(iii)	LXI H, 2500H SUB A, M HLT	Load H-L pair with 2500H Subtract the content of the memory, whose address is in H-L pair, to the content of the accumulator. Halt.

Q.14. Write and explain the execution of register-reference instruc

Or

Explain the control functions and microoperations for the reg reference instructions.

Ans. Register-reference instructions are identified by the control
$D_7 = 1$ and $I = 0$. In these in
are used to specify one of th
(0-11). The control functio
instructions are given in tab

Table 3.3 Execut

$D_7I'T_3$	=	r(common t
IR(i)	=	B_i [bit in IR
	r:	$SC \leftarrow 0$
CLA	rB_{11}:	$AC \leftarrow 0$
CLE	rB_{10}:	$E \leftarrow 0$
CMA	rB_9:	$AC \leftarrow \overline{AC}$
CME	rB_8:	$E \leftarrow \overline{E}$
CIR	rB_7:	AC ← shr
CIL	rB_6:	AC ← shl
INC	rB_5:	AC ← AC
SPA	rB_4:	If(AC(15
SNA	rB_3:	If(AC(15
SZA	rB_2:	If(AC = 0
SZE	rB_1:	If(E = 0)t
HLT	rB_0:	S ← 0(S is

These instructions are e
time variable T_3. Each contr
(designated by the symbol r
the bits in IR (0-11). All cont
assigning the symbol B_i to
instruction is completed at ti
control returns to fetch the n

First seven register-refe
circular shift, and increment
next four instructions cause a
stated condition is fulfilled
incrementing PC once again. T
as part of the control conditions. When the sign bit in AC (15) = 0, the AC
positive, and when AC (15) = 1, it is negative.

If all the flip-flops of the register are zero, the content of AC is
(AC = 0). The HLT instruction clears a start-stop flip-flop S and stops
sequence counter from counting. The start-stop flip-flop must be set manu
to restore the operation of the computer.

Q.15. Explain input-output instructions. *(R.G.P.V., Dec. 2011)*

Ans. Input and output instructions are required for transferring information to and from AC register, for checking the flag bits, and for controlling the interrupt facility. Input-output instructions contain an operation code 111 and …ed by the control when $D_7 = 1$ and $I = 1$. Remaining bits of the
Control functions and
given in table 3.4.

tions

uctions)	
instruction]	
	Clear SC
	Input character
	Output character
)	Skip on input flag
)	Skip on output flag
	Interrupt enable on
	Interrupt enable off

sition associated with timing
ction D_7IT_3 (designated by
of the bits in IR (6-11). By
can be represented by pB_j for
counter SC is cleared.

ree-address instructions ?
(R.G.P.V., May 2018)

stack organized computer
nd POP instructions require
nicates with the stack. The
as –

ADD		ToS ← (P + Q)
MUL		ToS ← (X + Y) * (P + Q)
POP	A	M [A] ← ToS

It is required to convert the expression into reverse polish notation to evaluate arithmetic expressions in a stack computer. The name zero address is

given to this type of computer because of the absence of an address fi the computational instructions.

One address instructions use an implied accumulator register for all manipulation. There is a need for a second register for multiplication division. However, here we will neglect the second register and assume the accumulator contains the result of all operations. The program

+ Y) * (P + Q) will be writte

```
LOAD     X
ADD      Y
STORE    T
LOAD     P
ADD      Q
MUL      T
STORE    A
```

T is the temporary memo

result.

In commercial computers

The program for A = (X + Y)

```
MOV      R1, X
ADD      R1, Y
MOV      R2, P
ADD      R2, Q
MUL      R1, R2
MOV      A, R1
```

Here, the first symbol li

source and the destination wh

Computers with three-ad

field to specify either a proces

for A = (X + Y) * (P + Q) wil

```
ADD    R1, X, Y
ADD    R2, P, Q
MUL    A, R1, R2
```

The advantage of the th

programs when evaluating ari

the binary coded instructions r

Q.17. *Discuss the following with examples –*

(i) Zero-address instructions

(ii) One-address instructions

(iii) Two-address instructions.

(R.G.P.V., Dec. 201

Ans. Refer to Q.16.

NUMERICAL PROBLEMS

Prob.1. If an instruction code has 4 bit opcode and 12 bit address field then

How many operations this code can perform ?

addressed ?

(R.G.P.V., June 2014)

96

unit with a capacity of

n code format consists of

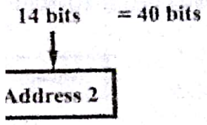
for the address part (no

cked in one memory word

in control unit. Formulate

ction for this computer.

(R.G.P.V., Dec. 2007)



structions –

m memory to IR and then

1.

2.

Prob.3. The memory unit of a computer has 256 K words of 32 bit each. The computer has an instruction format with 4 fields. An operation field a mode field to specify one of seven addressing modes, a register address field to specify one of 60 processor registers and a memory address. Specify the instruction format and the number of bits in each field if the instruction is in one memory word. *(R.G.P.V., June 2015)*

Sol. The instruction is of 32 bits long, in which the address bits are which are determined as

$$256\text{ K} = 2^8 \times 2^{10} = 2^{18}\text{ bytes}$$

Therefore,

Address bits =

To specify one of seven

$7 \le$

To specify one of 60 pr

Register bits are require

60 :

Opcode bits = Total bits

= 3

= 3

= 5

The instruction format is

5

Opcode

Prob.4. *A computer use each. A binary instruction code has four parts – an indirect a to specify one of 64 registers*

(i) How many bits code part and the address pa

(ii) Draw the instru bits in each part.

(iii) How many bits memory ?

Sol. (i) The instruction i 18, which are determined as –

$$256\text{ K} = 2^8 \times 2^{10} = 2^{18}\text{ bytes}$$

Therefore,

Address bits = 18 bits

Register code = 6 bits

Indirect bit = 1/25 bits

32 – 25 = 7 bits for opcode.

(ii) The instruction format is given below –

1	7	6	18
I	Opcode	Register	Address

ımetic statement –

ɔ)

(R.G.P.V., Dec. 2010)

thmetic statement by using

M[y]

M[z]

M[n]

M[o]

R2

M [S]

M [Q]

:

ithmetic statement by using

SUB	y	AC ← AC – M[y]
ADD	z	AC ← AC + M[z]
STORE	T	M[T] ← AC
LOAD	m	AC ← M[m]
MUL	n	AC ← AC * M[n]

Q.21. Write comparison between RISC and CISC.

(R.G.P.V., June 2

Ans.

S.No.	RISC	CISC
(i)	In RISC, the clock MHz in 1993.	In CISC, the clock
(ii)	Simple instructions cycle.	
(iii)	Average CPI is less	
(iv)	Few instructions may	
(v)	Instructions are exe ware.	

Q.22. Compare RISC a

Ans. Refer to Q.20 and

Q.23. Write short note

Ans. The addressing mod
are selected during program e
interpreting or modifying the
is actually referenced. Addre
the purpose of accommodatin

(i) To provide pro
facilities as pointers to mem
and program relocation.

(ii) To decrease th
instruction.

Q.24. Define implicit a

***Ans.* Register Address**
operands are in the general pu
of the registers in addition to

MOV A, B
78

In this example, the opcode for MOV A, B is 78 H. Besides the opera
to be performed the opcode else specifies the registers which contain d
The opcode 78 H can be written in binary form as 01111000. The first t
bits i.e., 01 are for MOV operation, the next three bits 111 are the binary co
for register A, and the last three bits 000 are the binary code for register B

Implicit Mode Addressing – There are certain instructions which operate on the content of the accumulator. Such instructions do not require the address of the operand. Examples are – CMA, RAL, RAR etc.

What must the address field of an indexed addressing mode
ected mode instruction?

(R.G.P.V., Dec. 2006)

field of the instruction
red in memory. Control
ss part to access memory

ield of the instruction be
The effective address in

struction + Content of

ress register is added to
dress. The index register

address field is removed
register indirect mode
d in an index addressing
ect mode instruction.

des supported by 8085

(R.G.P.V., June 2009)

is calculated in different

(R.G.P.V., Dec. 2010)

essing modes of a basic

(R.G.P.V., June 2011)

elp of example.

(R.G.P.V., June 2012)

s with an example.

(R.G.P.V., June 2013)

Or

Briefly explain all the addressing modes of computer instruction.

(R.G.P.V., June 2014)

Or

What are different addressing modes ? Explain each of them.

(R.G.P.V., June 2017)

Ans. The addressing mode of the instruction determines the way operands are selected during program execution. The addressing mode spec a rule for interpreting or modifying the address field of the instruction be the operand is actually referenced. Addressing mode techniques are use computers for the purpose o

provisions –

(i) To provide pr

such facilities as pointers to

data, and program relocation

(ii) To decrease th

instruction.

Types of Addressing M

(i) Inherent or In

specified in the instruction. C

the operand address and the

Example – All the 8-bi

instruction consists of only

with the format of fig. 3.7 (a

MPU	Mnemon
8085	CMA
6800	ABA

(ii) Immediate Ad

operand is contained as a pa

immediately following the c

addressing is fast in processin

address followed by the oper

Example – Almost all 8-

MPU	Mnemor
8085	ADI d
Z80	LD A,

(iii) Absolute Addressing – The most natural way of specifyi the memory address is to include it as a part of the instruction using th absolute (or extended or direct, as it is sometimes called) mode of addressin A typical instruction format along with the addressing mechanism is depict in fig. 3.7 (c).

Example – Because of its simplicity and flexibility this addressing mode is provided by all MPUs.

MPU	Mnemonic	Hex Code	Operation
8085	LDA addr	3A 33 01	(A) ← (1033)
			(A) ← (1033)

ress specified as a part of

address of the operand.

performed to obtain the

is slow in operation but

d address. The instruction

n most of the 8-bit MPUs.

Operation
A) ← (2000)

pose registers available in

igh speed memory. These

rs to the registers defining

can be used temporarily

nstructions using register

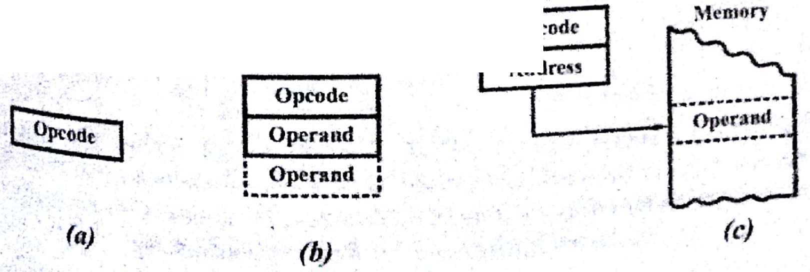
using absolute addressing

ister is much smaller than

extensive use of register

sses are provided as a part

peration
) + (r)
ts specify the register}



(a) (b) (c)

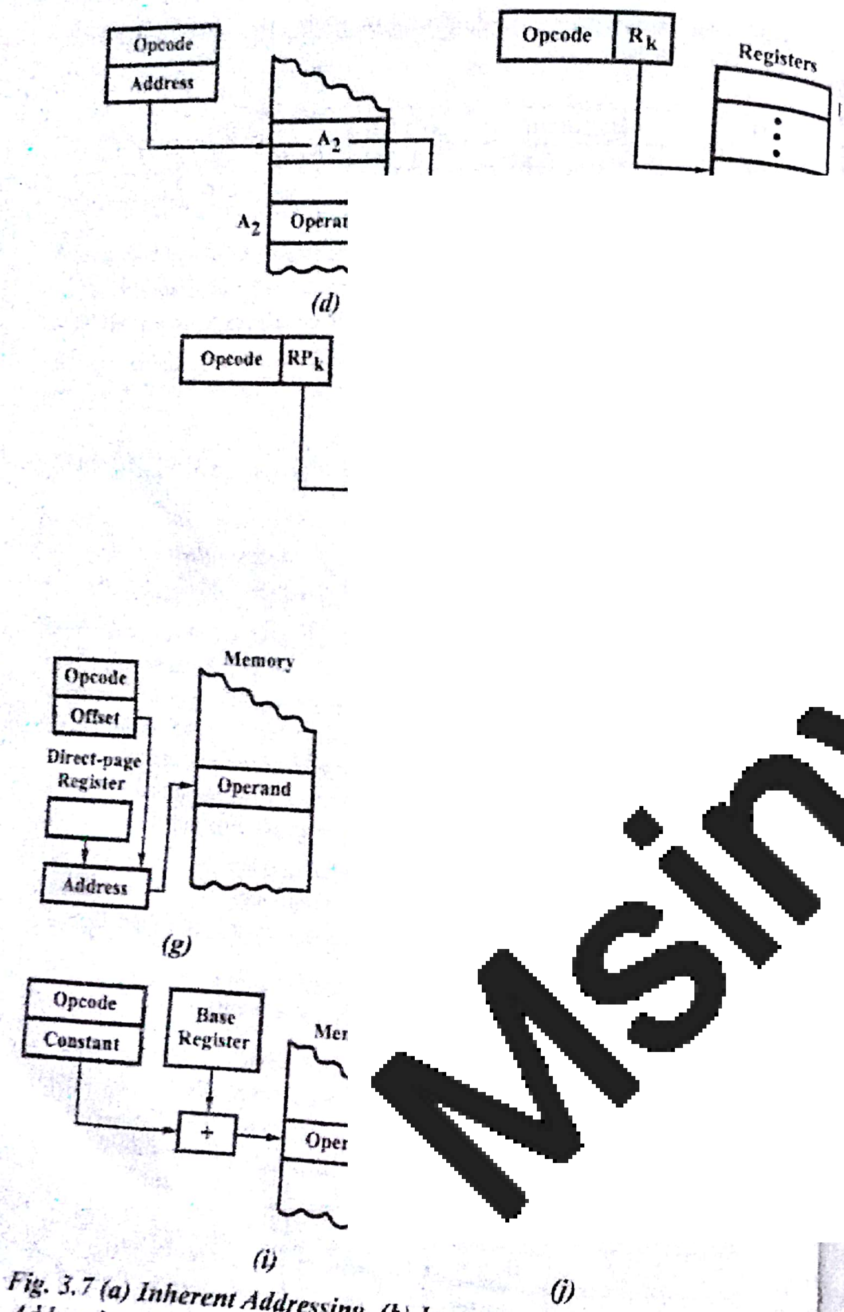


Fig. 3.7 (a) Inherent Addressing, (b) Immediate Addressing, (c) Absolute Addressing, (d) Indirect Addressing, (e) Register Addressing, (f) Register Indirect Addressing, (g) Paged Addressing, (h) Indexed Addressing, (i) Based Addressing, (j) Relative Addressing

(vi) Register-indirect Addressing – Conceptually same as indirect addressing, a register is used to contain the address of an operand. The register has to be loaded by the operand address before register-indirect addressing can be used. For 8-bit MPUs, the size of the address is double of the word

result, a register-pair is required to hold an address as depicted in

Operation
(A) ← (A) + ((CH)(L))
(reg) ← ((H)(L))
sss specifies register

memory space is logically
erally, the page size is 256
-register is used to hold the
et. The offset provides the
ng address of the page. The
f the page-register contents
order part of the address as

Operation
(A) ← (A) + (0033)
(A) ← (A) + (00F9)

ode, a fixed base address
. Then, an offset from the
own as index register. The
it with the contents of the

				Operation
				(A) ← ((X) + 174F)
6809	LDA	23, X	A6 88 23	(A) ← ((X) + 23)

(ix) Based Addressing – Another addressing scheme which uses a register to generate effective address is known as based addressing. The address generation process is exactly the same as in the case of indexed addressing; the contents of a register called based register is added with the offset provided

as a part of the instruction to obtain the effective address as depicted fig. 3.7 (i).

Example –

MPU	Mnemonic	Hex Code	Operation
Z80	LD A, (		
6800	ADD A$15		

(x) Based-Indexe
high flexibility by using two
generating the effective add
index register are added tog
indexed addressing permits
and an offset to an item of a

Example –

MPU	Mnemo
6809	LDA B,

(xi) Indexed Indire
address generated by indexe
There are two possibilities as

(a) Pre-indexe
the base address to form an in
is the effective address.

(b) Post-indexe
used as an indirect address; th
address to get the effective ad

Example –

MPU	Mnemonic		
6502	LDA ($04,		
6809	LDA 23, x	A6 98 23	(A) ← ((x) + 23)

(xii) Relative Addressing – In this mode of addressing, the effective address is generated by adding the current value of the program counter with a fixed signed displacement available as a part of the instruction as depicted in fig. 3.7 (j).

Example –

MPU	Mnemonic	Hex Code	Operation
Z80	JP Z,02	28 08	If Z = 0 continue, if Z = 1 (PC) ← (PC) + 08
			Z = 0 continue, Z = 1 (PC) ← (PC) + 08

inter register is used to specify
Instruction length is shortest,
ster or a memory location is
is automatically incremented
ng stack addressing. In PUSH
can be stored into a memory
ented by one or two. Likewise,
location pointed by the (SP)
the SP is incremented by one

Operation
SP) – 1) ← (rh),
SP) – 2) ← (r*l*),
P) ← (SP) – 2
SP)) ← (A),
P) ← (SP) – 1

a transfer.
(R.G.P.V., June 2015)

nsfer ? Explain each mode in
(R.G.P.V., Dec. 2013)

nsfer –
lriven (iii) Direct memory access.
(R.G.P.V., June 2011)

Or

Write three modes of data transfer and explain any one of them.
(R.G.P.V., June 2014)

Or

Explain the different modes of data transfer between the central computer and I/O devices.
(R.G.P.V., Dec. 2015)

Or

Explain the three ways of data transfer to and from peripherals.
(R.G.P.V., May 20

Ans. There are three modes of transferring data between the comp
and I/O device –

(i) Programmed I
(ii) Interrupt initia
(iii) Direct memor

Programmed I/O operat
computer program. Each data
program. Generally, the trans
Also other instructions are ne
and CPU. Transferring data ur
of the peripheral by the CPI
required to monitor the inter

In programmed I/O meth
I/O unit indicates that it is ready
because it keeps the processo
that the device is ready for da
computer. After detecting the e
the task it is processing branch
and then comes back to the tas

In programmed I/O, tran
DMA (direct memory access
memory unit through the m
supplying the interface with
required to be transferred and
transfer is made, DMA reque

If request is granted by
directly into memory. The CP
permit the direct memory I/O
than processor, I/O memory
access to memory.

Q.28. What do you mean

Ans. Refer to Q.27.

Q.29. Explain in short programmed I/O and interrupt initiated I/O.
(R.G.P.V., June 2005)

Or

Write short notes on memory mapped I/O and I/O mapped I/O.
(R.G.P.V., Dec. 2006, 2009)

Ans. **Programmed I/O** – Programmed I/O is used in every computer for
ontrolling I/O operations. Programmed I/O needs that all I/O operations be
xecuted under the direct control of the CPU. It means that every data-transfer
peration involving an I/O device needs the execution of an instruction by the
lly, the transfer is between two programmable registers – one a
device. The I/O device does
ansfer from an I/O device to
structions, including an input
ice to the CPU and a store
memory unit.

PU, main memory, and I/O
. Address lines of the system
can also be used to choose
bus via an I/O port, which
ata register, hence making it

mory-mapped I/O is used to
ice to I/O ports. A memory
fetched from or stored at
ion if X is made the address
ore instructions are used to
pecial I/O instructions are
th this type of I/O addressing
WRITE, which are activated
nce instruction, are used to
ransfer.

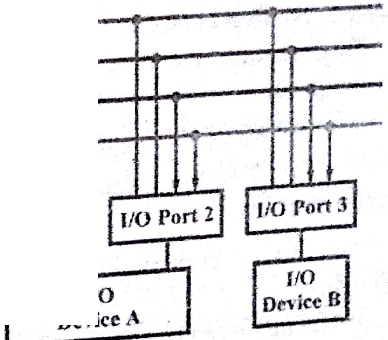


Fig. 3.8 Programmed I/O with Shared Memory and I/O Address Space (Memory-mapped I/O)

In the organization depicted in fig. 3.9, sometimes known as I/O-mapped I/O, the memory and I/O address spaces are separate. In this scheme, a memory-referencing instruction activates the READ M or WRITE M control line which

does not affect the I/O devices. The CPU must execute separate I/O instruc
to activate the READ IO and WRITE IO lines, which cause a word
transferred between the addressed I/O port and the CPU. An I/O device
memory location can consist of the same address bit pattern without con

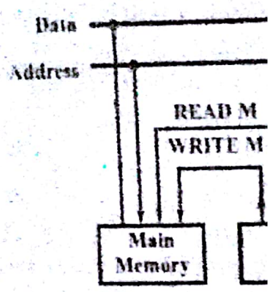


Fig. 3.9 Progr
I/O A

When the CPU exe
addressed I/O port is ex
the I/O device must tran
specified period. The CP
of information or an inde
transfer is carried out o
the CPU can be program
I/O data transfer. Frequen
that the I/O device mak
setting a flip-flop conne

To determine the s
following steps –

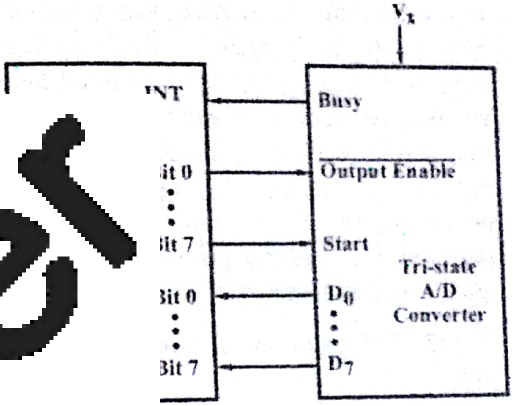
(i) Read the

(ii) Test the s
transferring data.

(iii) If not read
transfer.

Interrupt Initiated I/O – Interrupt I/O is a device-initiated I/O transf
The external device is connected to a pin known as the interrupt (INT) pin
the processor chip. If the device requires an I/O transfer with the compu
then it activates the interrupt pin of the processor chip. Generally, the comp
completes the current instruction and saves at least the contents of the curr
program counter in the stack.

Then, the computer automatically loads an address into the program counter to branch to a subroutine-like program known as the interrupt service routine. This program is written by the
er. The external device desires
o execute this



) Computer A/D Converter
rface via Interrupt I/O

ts – external interrupts, traps or

e interrupt pins of the computer
xternal interrupts can further be
skable. A maskable interrupt is
tions such as EI or DI. The
mpared to the maskable interrupt.
are activated at the same time,
nterrupt first. The nonmaskable
interrupt.

ernally by exceptional conditions
of an illegal opcode. Traps are
rupts.

rupts or system calls. When one
essor is interrupted or serviced
oftware interrupt instructions are

following –
(i) Programmed I/O (ii) Interrupt initiated I/O (iii) DMA
Justify your answer.

Ans. There are three techniques for I/O operations. With programmed I/O, data are exchanged between the processor and the I/O module. The processor executes a program that gives it direct control of the I/O operation, including sensing device status, sending a read or write command, and

transferring the data. When the processor issues a command to the module, it must wait until the I/O operation is complete. If the processor faster than the I/O module, this is wasteful processor time. With inter driven I/O, the processor issues an I/O command, continues to exec other instructions, and is interrupted by the I/O module when the

finished its work. With both
responsible for extracting d
data in main memory for in
access (DMA). In this mod
data directly, without proce
most efficient technique.

Q.31. Explain the draw
I/O.

Ans. Interrupt-driven I/O
I/O, still demands the active
between I/O module and mer
through the processor. Thu
drawbacks –

(i) The speed wit
device limits the I/O transfer

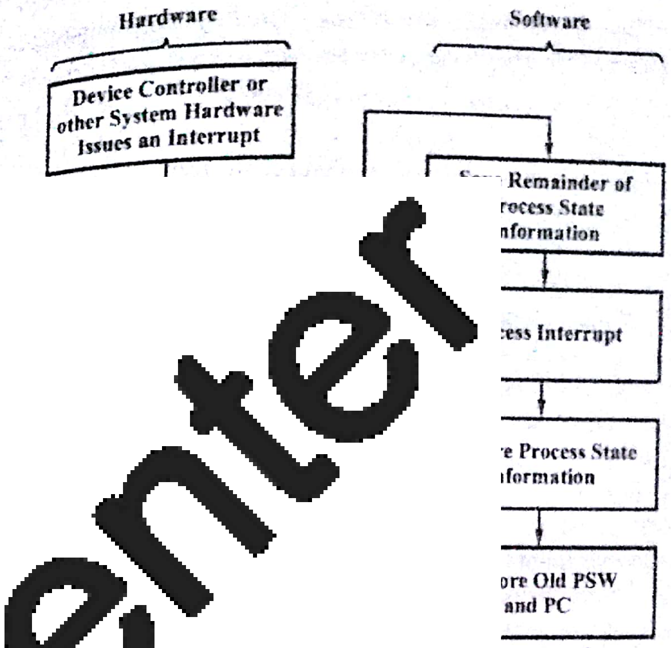
(ii) The processor i
instructions must be executed

Q.32. How is interrupt dr
completely how the various si

Ans. The problem with pr
a long time for the I/O module
transmission of data. The proce
the status of the I/O module.
entire system is severely degra

An alternative is for the pr
and then go on to do some oth
the processor to request servic
processor. The processor then execute the data transfer, as before, and
resume its former processing.

Interrupt is more efficient than programmed I/O because it elimin
eedless waiting. However, interrupt I/O still consumes a lot of proces
ime because every word of data that goes from memory to I/O module
om I/O module to memory must pass through the processor.



essing
, the following sequence

to the processor.
on and responding to the

determines that there is
e that issue the interrupt.
to transfer control to the
ume the current program
required is the status of
be executed.
counter with the entry
espond to this interrupt.
At this point, the program counter and PSW relating to the interrupted program have been saved on the system stack.

(vii) The interrupt handler next processes the interrupt.

(viii) Now saved register values are retrieved from the stack and restore to the register.

(ix) The final act is to restore the PSW and program counter value from the stack.

Q.33. What is the basic advantage of using interrupt-initiated d transfer over transfer under program control without an interrupt ?
(R.G.P.V., Dec. 2008, June 20

Ans. Refer to Q.32.

Q.34. With the help of suitable flowchart explain the interrupt i data transfer operation. I

Ans. The problem wi
a long time for the I/O m
transmission of data. The p
the status of the I/O modu
entire system is severely c
An alternative is for t
and then go to do some ot
the processor to request s
processor. The processor i
resumes its former proces
Let us consider how
module. For input, the I/O m
The I/O module then proc
Once the data are in th
register, the module signal
the processor over a control
then waits until its data are
processor. When the requ
module places its data on th
then ready for another I/O
the processor's point of vie
input is as follows. The pr
READ command. It then g
something else. At the end o
cycle, the processor check
When the interrupt from t
occurs, the processor saves
the current program and
interrupt. In this case, the p
the word of data from the I/O module and stores it in memory. It then restores the context of the program it was working on and resumes execution.

Fig. 3.12 shows the use of interrupt I/O for reading in a block of data.

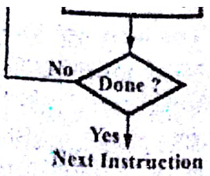


Fig. 3.12 Interrupt-driven I/O

Advantages and Disadvantages – Interrupt I/O is more efficient than programmed I/O because it eliminates needless waiting. However, interrupt I/O still consumes a lot of processor time, because every word of data that goes from memory to I/O module or from I/O module to memory must pass the processor.

EMS

A relative mode branch type
equivalent to decimal 750.
l 500.

elative address field of the

lue in binary using 12 bits.

' after the fetch phase and
hat the binary value in PC
n equal to the binary value

(R.G.P.V., Dec. 2009)

51

using 12 bits is

ause the relative address of

$$500 = 000111110100$$

$$\begin{aligned} PC &= 751 = 001011101111 \\ RA &= \underline{-251} = \overset{+}{\underline{111100000101}} \\ EA &= 500 = 000111110100 \end{aligned}$$

Prob.7. *What is addressing mode ? An instruction is stored at loca 300 with its address field at location 301. The address field has the va 400. A processor register R1 contains the number 200. Evaluate the effect address if the addressing mode of the instruction is –*

(i) Direct
(ii) Immediate
(iii) Register Indi
(iv) Relative
(v) Index with R.

Sol. **Addressing Mode**

PC

R1 = 20

Now,
Effective address –
(i) Direct – 400
(ii) Immediate – 3
(iii) Relative – 302
(iv) Register indire
(v) Indexed – 200

Prob.8. *At memory ad stored with a mode bit as a stored is 500. At location numbers are stored at diffe*

Memory *(Add*	
39	
40	
50	
60	*900*
702	*325*
800	*300*

If the content of PC is 200, while the contents of register R1 is 400 XR register is 100. If all the numbers and address are in decimal numbe find out contents of AC and effective address for the following addressin modes –

(i) Direct address
(ii) Indirect address
(iii) Relative address
(iv) Indexed address
Register indirect addressing mode.

(R.G.P.V., Dec. 2005)

Table 3.5

	Effective Address	*Content of AC*
	500	800
s	800	300
s	702	325
s	600	900
t	400	700

er than an address, therefore s case is 201. In the *indirect* ry at address 500. So, the . In the ***relative mode***, the erand is 325. In the ***index*** 500 = 600 and the operand

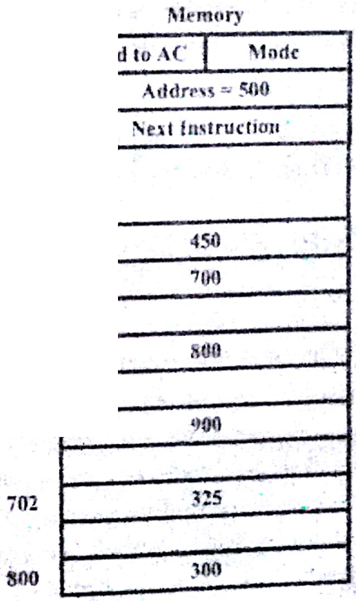


Fig. 3.13

is 900. In the ***register mode***, the operand is in R1 and 400 is loaded into
There is no effective address in this case. In the ***register indirect mode***,
effective address is 400, equal to the content of R1 and the operand lo
into AC is 700. Table 3.5 lists the values of the effective address and
operand loaded into AC for the addressing modes.

PRIORITY

Q.35. What do you m
how does the processor de

Ans. Program interru
a currently running progr
external or internal generat
after the service program i

The hardware procedu
execution of a subroutine
execute cycle (when the in
the program counter, the c
certain status conditions. T
and the beginning address
counter. The beginning ad
hardware. The service rout
and proceed to service it.

Q.36. What is meant
establishing the priority o

Explain priority inter
I/O.

Write short note on pr

Ans. Priority interrupt
various sources to determine
or more requests arrive simultaneously. The system can also determine whic
conditions are allowed to interrupt the computer while another interrupt i
being serviced. Higher-priority interrupt levels are allocated to request which
if delayed or interrupted, could have serious results. Devices with high spee
transfers are given high priority and slow devices receive low priority. If tw

devices interrupt the computer at the same time, then computer services the device with the higher priority first.

The priority of simultaneous interrupts can be established by software or hardware. A ***polling*** method is used to recognize the highest-priority source
In this method, one common branch address is used for
errupts starts at the branch
nce. The priority of each
re tested. First the highest-
s on, control branches to a
xt-lower-priority source is
ne for all interrupts has a
ce and branches to one of
routine reached belongs to
h interrupted the computer.
if there are many interrupts,
available to service the I/O
t unit can be used to speed

rrupt requests from many
ts has the highest priority,
sed on this determination.
to access its own service
cause all the decisions are
nit. The establishment of
rial or a parallel connection
the ***daisy chaining method.***

? (R.G.P.V., June 2016)

ity interrupt over a non-
interrupt without a mask
(R.G.P.V., June 2008)

devices are attached to the
e an interrupt request. The
e source of the interrupt.
is also the possibility that several sources will request service simultaneously. The priority interrupt system establishes a priority over the various sources to determine which condition is to be serviced first when two or more requests arrive simultaneously.

It is not possible to have a priority interrupt without a mask register.

Q.39. Explain the process of handling an interrupt that occurs du the execution of a program, with the help of an example.

(R.G.P.V., Dec. 2009, 20

Ans. The way that the interrupt is handled by the computer can be expl
by means of the flowchart of
computer. When R = 0, the c
the execute phase of the inst
0, it indicates that the progra
continues with the next inst
bits. If both flags are 0, it ind
are ready for transfer of inf
next instruction cycle. If eith
1. At the end of the execute
it goes to an interrupt cycle

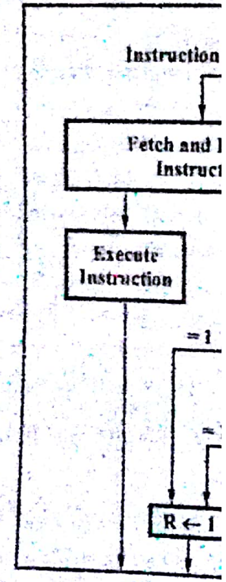
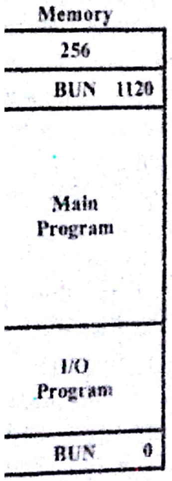


Fig. 3.14

The interrupt cycle is a
return address operation. Tl
address available in PC is store
specific location where it can be found later when the program returns to th instruction at which it was interrupted. This location may be a process register, a memory stack, or a specific memory location. Here, we choose th memory location at address 0 as the place for storing the return addres Control then inserts address 1 into PC and clears IEN and R so that no mo interruptions can occur until the interrupt request from the flag has been service

An example that shows what happens during the interrupt cycle is shown in fig. 3.15. Suppose that an interrupt occurs and R is set to 1 while the control is executing the instruction at address 255. At this time, the return ress 256 is in PC. The programmer has previously placed an input-output
memory starting from address 1120 and a BUN 1120
(a).
ds that R = 1, it proceeds
tored in memory location
ing of the next instruction
address 1 since this is the
1 causes the program to
lress 1120. This program
hen transfers the required
struction ION is executed
where it was interrupted.



r Interrupt Cycle

rupt Cycle

n interrupt ? Draw the

(R.G.P.V., June 2015)

Ans. Refer to Q.35 and Q.39.

Q.41. Write short note on parallel priority interrupt.

Ans. In this method, a register whose bits are set separately by the interrupt signal from each device is used. Priority is established according to the position of the bits in the register. In addition to the interrupt register, the circuit may include a mask register which is used to control the status of each interrupt

request. The mask register can be programmed to disable lower-prio interrupts while a higher-priority device is being serviced. In addition, it provide a facility, that permits a high priority device to interrupt the CPU wh a lower-priority device is being serviced.

Fig. 3.16 shows the priority logic for a system of four interrupt sou

It has an interrupt register
and cleared by program inst
disk because it is a high-sp
followed by a character r
same number of bits as the
interrupt register. It i
possible to set or reset any
bit in the mask register by
means of progran
instructions. Each interrup
bit and its corresponding
mask bit are connected to
an AND gate to generate the
four inputs to a priority
encoder. In this manner, ar
interrupt is identified only i
its corresponding mask bi
is set to 1 by the program
The priority encoder
produces two bits of the
vector address, that is
transferred to the CPU.

When an interrupt tha
the encoder sets an intern
IEN can be set or cleared l
the interrupt system. Outp
interrupt signal for the CPU
CPU enables the bus buffer
is kept into data bus.

Q.42. Write short note

Explain daisy chaining priority interrupt. *(R.G.P.V., Dec. 20*

Or

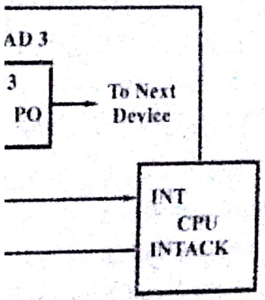
Briefly explain daisy-chaining priority method of interrupt.
(R.G.P.V., June 20

Or

Explain daisy-chaining priority for data transfer. *(R.G.P.V., June 20*

Ans. In the daisy-chaining method of establishing priority, all devices that request an interrupt are connected serially. The device with the highest priority is kept in the first position, followed by lower-priority devices up to the device with the lowest priority, which is kept last in the chain. Fig. 3.17 shows this method of connection between three devices and the CPU. The interrupt request

logic connection. When any
te, the interrupt line goes to
out in the CPU. In case no
n the high-level state and no
alent to a negative logic OR
line, the CPU responds to an
ice 1 at its PI (priority in)
next device through the PO
uesting an interrupt. When
knowledge signal from the
it proceeds to insert its own
or the CPU to use during the



Interrupt

0 in its PO output to inform
1 has been blocked. A device,
ill intercept the acknowledge
ice does not have pending
t device by placing a 1 in its
0 is the one with the highest
vice places its VAD on the

ous. In the daisy-chain arrangement, the highest priority is given to the device which receives the interrupt acknowledge signal from the CPU. The farther the device is from the first position, the lower is its priority.

The internal logic that must be included within each device when connected in the daisy-chaining arrangement is shown in fig. 3.18. The device sets its RF

request. The mask register can be programmed to disable lower-prio interrupts while a higher-priority device is being serviced. In addition, it provide a facility, that permits a high priority device to interrupt the CPU w a lower-priority device is being serviced.

Fig. 3.16 shows the priority logic for a system of four interr

It has an interrupt registe
and cleared by program in
disk because it is a high-s
followed by a character
same number of bits as t
interrupt register. It
possible to set or reset a
bit in the mask register l
means of progra
instructions. Each interru
bit and its correspondi
mask bit are connected
an AND gate to generate t
four inputs to a priori
encoder. In this manner, a
interrupt is identified only
its corresponding mask b
is set to 1 by the progra
The priority encod
produces two bits of t
vector address, that
transferred to the CPU.

When an interrupt t
the encoder sets an inter
IEN can be set or cleared
the interrupt system. Out
interrupt signal for the CP
CPU enables the bus buff
is kept into data bus.

Q.42. Write short no

Explain daisy chaining priority interrupt. *(R.G.P.V., Dec. 20*

Or

Briefly explain daisy-chaining priority method of interrupt.

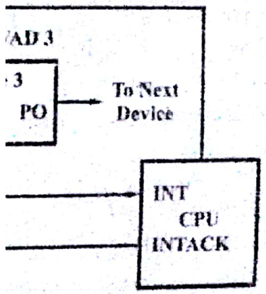
(R.G.P.V., June 20

Or

Explain daisy-chaining priority for data transfer. (R.G.P.V., June 20

Ans. In the daisy-chaining method of establishing priority, all devices that request an interrupt are connected serially. The device with the highest priority is kept in the first position, followed by lower-priority devices up to the device with the lowest priority, which is kept last in the chain. Fig. 3.17 shows this connection between three devices and the CPU. The interrupt request

logic connection. When any
ate, the interrupt line goes to
put in the CPU. In case no
in the high-level state and no
valent to a negative logic OR
line, the CPU responds to an
vice 1 at its PI (priority in)
next device through the PO
questing an interrupt. When
cknowledge signal from the
, it proceeds to insert its own
or the CPU to use during the



Interrupt

0 in its PO output to inform
l has been blocked. A device,
ill intercept the acknowledge
ice does not have pending
t device by placing a 1 in its
0 is the one with the highest
vice places its VAD on the
ous. In the daisy-chain arrangement, the highest priority is given to the device which receives the interrupt acknowledge signal from the CPU. The farther the device is from the first position, the lower is its priority.

The internal logic that must be included within each device when connected in the daisy-chaining arrangement is shown in fig. 3.18. The device sets its RF

flip-flop when it wants to interrupt the CPU. Output of the RF flip-flop through an open-collector inverter, a circuit which provides the wired for the common interrupt line. When PI = 0, both PO and the enable VAD are equal to 0, irrespective of the value of RF. When PI = 1 and RF then PO = 1 and the vector address is disabled. This condition pas

acknowledge signal to th

the device is active. This

by placing a 0 in PO. He

vector address. After a s

the CPU has received th

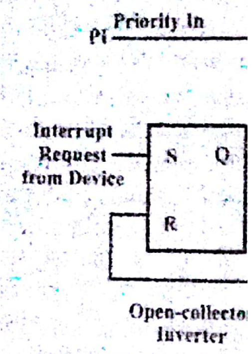


Fig. 3.18 One Sta

Q.43. Explain polli
priority interrupt.

Ans. **Polling** – Refe

Daisy-chaining Met

Q.44. Why is priorit
do the different priority s

Ans. Refer to Q.36, (

Q.45. What do you n
handling techniques.

Ans. **Interrupt** – Re

Interrupt Handling ques – The various interrupt han
techniques are –

(i) ***Polling*** – Refer to Q.36.
(ii) ***Parallel Priority Interrupt*** – Refer to Q.41.
(iii) ***Daisy-chaining Priority*** – Refer to Q.42.

DMA, INPUT-OUTPUT PROCESSOR (IOP)

Q.46. Write short note on DMA. ***(R.G.P.V., June 2005, 2006)***

(R.G.P.V., June 2016)

e device such as magnetic e CPU. Removing the CPU nanage the memory buses ransfer technique is known and has no control of the e transfer directly between es over the buses.

in in detail how this is
(R.G.P.V., June 2013)

iety of ways. One common ses through special control ie CPU that facilitate the by the DMA controller to When this input is active, nstruction and places the ies into a high-impedance pen circuit, which means a logic significance. The he external DMA that the originated the bus request emory transfers without ie transfer, it disables the ikes control of the buses.

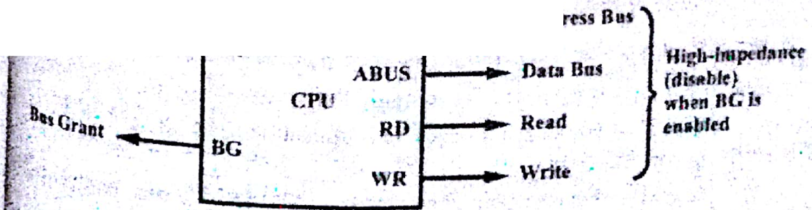


Fig. 3.19 CPU Bus Signals for DMA Transfer

When the DMA takes control of the bus system, it communicates d
with the memory. The transfer can be made in several ways. In DMA
transfer, a block sequence consisting of a number of memory words is tran
in a continuous burst while the DMA controller is master of the memory
This mode of transfer is needed for fast devices such as magnetic dis
data transmission cann
transferred. An alterna
controller to transfer on
of the buses to the CPU
cycle to allow the direc

Q.48. Differentiat
transfer.

Ans. With prograt
and the I/O module. T
control of the I/O opera
write command, and tr

In direct memory
exchange data directly,

Q.49. Why are r
bidirectional ? Under v
inputs ? Under what co

Ans. When the CI
and write lines are used

When the DMA c
write lines are used as

Thus, the read and

Q.50. Why does D.
memory transfer ? Exp

Ans. The CPU can
without any damage oc
transfers data from a dev
to flow, so loss of data

Q.51. Explain the r

Ans. The main functions of a typical DMA controller are as follows
(i) The I/O devices request DMA operation through the DMA req
line of the controller chip.

(ii) The controller chip activates the CPU HOLD pin, requesting
CPU to release the bus.

(iii) The processor sends HLDA (hold acknowledge) back to the DMA
controller, indicating the bus is disabled. The DMA controller places the current
value of its internal register, such as the address register and counter, on the
system bus and sends a DMA acknowledge to the peripheral device. The
completes the DMA transfer.

itialised I/O and direct
(R.G.P.V., Dec. 2011)

an I/O command and then
rdware to signal the end of
y access a specialized I/O
ve a large block of data.

king principle of DMA
(R.G.P.V., Dec. 2008)

controller ? How does
diagram.
(R.G.P.V., Dec. 2009)

iple of DMA controller.
(R.G.P.V., June 2012)

1. (R.G.P.V., Dec. 2014)

(R.G.P.V., June 2017)

ler is shown in fig. 3.20.
and control lines. In the

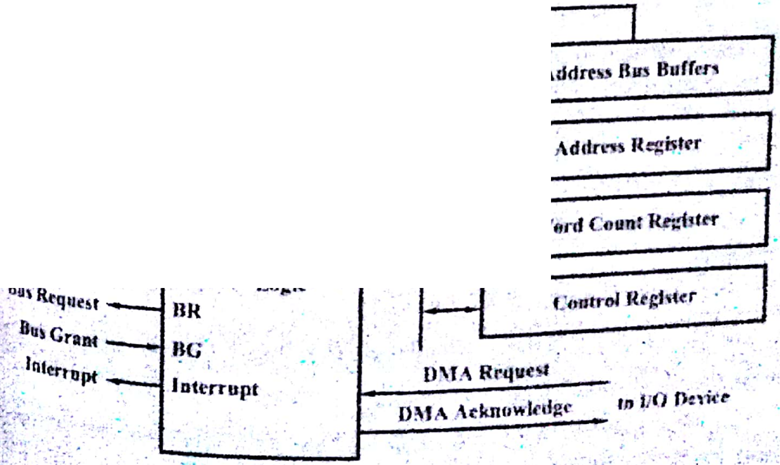


Fig. 3.20 Block Diagram of DMA Controller

DMA, the registers are chosen by the CPU through the address bus by enab
the DS (DMA select) and RS (register select) inputs. The RD (read) and
(write) inputs are bidirectional. If the BG (bus grant) input is 0, then the C
can communicate with the DMA registers through the data bus to read
or write to the DMA registers. If BG = 1, then CPU has relinqui
and the DMA can comm
address in the address bus
prescribed handshaking pr
peripheral through the req

DMA controller has thr
and a control register. To spe
contains an address. The ad
Address register is incremen
number of words to be trans
is decremented by one after
mode of transfer is specified
to the CPU as I/O interface re
DMA registers under progra

First, the DMA is init
continues to transfer data b
block is transferred. Basicall
I/O instructions, which inclu

The CPU initializes the
the data bus –

(i) The starting
available (for read) or whe
(ii) The word cou
(iii) Control to sp
(iv) A control to

The starting address is
stored in the word count re
register. Once the DMA is i
DMA unless if receives an
words have been transferre

Q.54. What is a DMA
principle.

Ans. Refer to Q.46, Q.47 and Q.53.

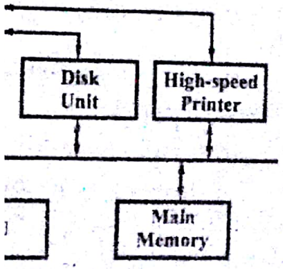
Q.55. What is a DMA controller ? How it transfers data in a comp
system ? Write in short.
(R.G.P.V., June 20

Or

Explain how the data is transferred between memory and I/O de
using DMA controller.
(R.G.P.V., Dec. 20

Ans. When large blocks of data need to be transferred at high speed, a special control unit may be provided to enable transfer of a block of data directly between an external device and the main memory, without continuous
...tion by the CPU. This approach is called ***direct memory address*** or
registers, one for generating
k of the word count. A third
he function to be performed.
drives, other registers may
controller registers must be
gram control, they should be
sses, as in the case of any

ntroller used in conjunction
d printer. In such a case, the
. The registers required to
on are duplicated, so that
connection is also provided



-channel DMA Controller

Vrite).

ndently to implement the
sk unit to synchronise its
of DMA transfer, this fact
troller. The status register
transfer occurred correctly
encountered.

While the DMA transfer is occurring, the program that requested the transfer cannot continue. However, the CPU can be used to execute another program. When the DMA transfer is completed, the CPU may switch back to the program that requested the transfer. It is the responsibility of the operating system to suspend the execution of one program and to start another. It is also the operating system which initiates the DMA operation when requested to do

so by a program. After the transfer is completed, the DMA controller info
the CPU by means of a control signal on the bus, known as the inter
request signal. The controller activates this signal at the same time it sets
Ready bit in its status register.

It is important to note that a conflict situation may arise if both th
a DMA controller try to acc
this conflict, a special circuit
activities of all devices requ
priority system. Memory a
interwoven with top priorit
speed peripherals, like disk
CPU generates the majority
be regarded as "stealing" me
technique is generally know
may be given exclusive acc
without interruption. This is

Q.56. What are the di
actual process of direct me

What are the differen
principle of DMA.

Ans. There are three b

(i) Block-transf
from the computer to tran
device. The CPU has no
During this time, the CPU c
bus. This method is popular
of data can be transferred.

(ii) Cycle Steali
memory and an I/O device
CPU is generated by ANDi
has the same frequency as

(iii) Interleaved
system bus when the CPU is not using it. For example, the CPU does no
the bus while incrementing the program counter or performing an ALU operatio
The DMA controller chip identifies these cycles and allows transfer of da
between the memory and I/O device. Data transfer take place over a period
time for this method.

Also, refer to Q.55.

Q.57. What are different modes of data transfer ? Explain the DMA controller with block diagram. What is meant by block transfer ?

(R.G.P.V., Dec. 2012)

Ans. **Modes of Data Transfer** – Refer to Q.56.

Block Diagram of DMA Controller – Refer to Q.53.

fer in computer system and

by the data bus and address
It activates the RS line and
n initialize the DMA by the
ral device and the memory.
nd. The DMA controller is
ts a DMA request, informing
with its BG line, informing
MA puts the current value

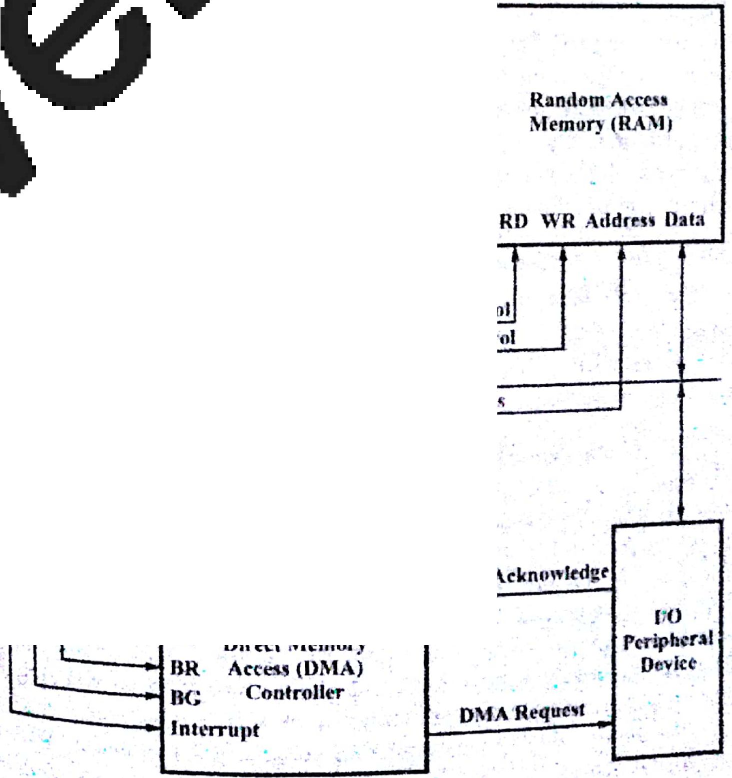


Fig. 3.22 DMA Transfer in a Computer System

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer